**Лабораторно упражнение №5**

**Възможности на Python за разработване на мрежови приложения. Обработване на данни в HTTP протокол, обработване на уеб страници и файлове с urllib, обработване на HTML документи**

Основни понятия:

* *BeautifulSoup* – това е библиотека на Python за анализ и извличане на данни от HTML документи. BeautifulSoup код може да се изтегли от [www.crummy.com](http://www.crummy.com).
* *port* - число, което обикновено показва с коя услуга (service) се свързвате, когато правите сокет връзка със сървър. Например, услугата уеб сървър обикновено използва порт 80, а услугата електронна поща използва порт 25.
* *scrape* - когато програма се преструва на уеб браузър и извлича уеб страница, след което разглежда съдържанието на уеб страницата.
* *socket* - мрежова връзка между две приложения, при която приложенията могат да изпращат и получават данни във всяка посока.
* spider – действа като уеб търсачка за извличане на страници.

При разработване на мрежови приложения е необходимо да се добави библиотеката socket

**Import socket**

**1. HyperText Transfer Protocol – HTTP**

Мрежовият протокол, който захранва мрежата, всъщност е доста прост и има вградена поддръжка в Python. Сокетите спомагат за лесно създаване на мрежови връзки и извличане на данни.

Socket е файл, който осигурява двупосочна връзка между две програми. В един и същ сокет могат да се записват и прочитат данни.

Ако се запишат данни в сокет, те се изпращат до приложението в другия край на сокета. Ако се чете от сокета, се получават данните, които приложението в другият край на сокета е изпратило.

Протоколът е набор от точни правила, които определят кой трябва да се изпрати/получи пръв.

**2. Най-простият уеб браузър**

Може би най-лесният начин да се покаже как работи HTTP протоколът е да се напише много проста програма на Python, която осъществява връзка с уеб сървър и следва правилата на HTTP протокола, за да поиска документ от сървъра и да го върне обратно.

**import** socket  
mysock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
mysock.connect((**'data.pr4e.org'**, 80))  
cmd = **'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'**.encode()  
mysock.send(cmd)  
**while True**:  
 data = mysock.recv(512)

print(data.decode(),end=**''**)

**if** len(data) < 1:  
 **break**

mysock.close()

Първо програмата прави връзка с порт 80 на сървъра www.py4e.com. Тъй като нашата програма играе ролята на „уеб браузър“, HTTP протоколът казва, че трябва да изпратим командата GET, последвана от празен ред. \ r \ n означава EOL (края на реда), така че \ r \ n \ r \ n не означава нищо между две EOL последователности.

Това е еквивалент на празен ред.

След като се изпрати празен ред, следва цикъл, който приема данни от 512 символа от сокета и ги отпечатва, докато няма повече данни за четене (т.е. recv () връща празен низ).

**Примерен изход**

The program produces the following output:

HTTP/1.1 200 OK

Date: Wed, 11 Apr 2018 18:52:55 GMT

Server: Apache/2.4.7 (Ubuntu)

Изходът започва със заглавни редове, в които уеб сървърът изпраща описание на документа.

Например, Content-Type показва, че документът е обикновен текстов документ (text/plain).

След като сървърът ни изпрати описанията, той добавя празен ред, за да посочи края на описанията и след това изпраща действителните данни на файла romeo.txt.

Този пример показва как се осъществява мрежова връзка със сокети на ниско ниво.

Сокетите могат да се използват за комуникация с уеб сървър, с пощенски сървър или много други видове сървъри. Всичко, което е необходимо, е да се намери документ, който описва протокола, и да се напише кода, за да се изпратят и получат данните в съответствие с протокола.

Тъй като протоколът, който се използва най-често е уеб протокол HTTP, Python има специална библиотека, която поддържа HTTP протокола за извличане на документи и данни през мрежата.

Едно от изискванията за използване на HTTP протокола е необходимостта да се изпращат и получават данни като байтови обекти вместо низове. В предходния пример методите encode() и decode() преобразуват низове в байтови обекти и обратно.

Следващият пример използва b '' нотация, за да посочи, че променлива трябва да се съхранява като байтов обект. Действието на функцията encode() е еквивалентно на нотацията b ''.

>>> b'Hello world'

b'Hello world'

>>> 'Hello world'.encode()

b'Hello world'

**3 Извличане на изображение през HTTP**

В горния пример се извлече обикновен текстов файл, който имаше нови редове и данните се изведоха на екрана. Може да се използва подобна програма за извличане на изображение в целия HTTP. Вместо да се извеждат данните на екрана, данните може да се натрупват в низ и след това да се запазват във файл, както следва:

**import** socket  
**import** time  
HOST = **'data.pr4e.org'**PORT = 80  
mysock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
mysock.connect((HOST, PORT))  
mysock.sendall(**b'GET http://data.pr4e.org/cover3.jpg HTTP/1.0\r\n\r\n'**)  
count = 0  
picture = **b""  
while True**:  
 data = mysock.recv(5120)  
 **if** len(data) < 1: **break** *#time.sleep(0.25)* count = count + len(data)  
 print(len(data), count)  
 picture = picture + data  
mysock.close()  
*# Look for the end of the header (2 CRLF)*pos = picture.find(**b"\r\n\r\n"**)  
print(**'Header length'**, pos)  
print(picture[:pos].decode())  
*# Skip past the header and save the picture data*picture = picture[pos+4:]  
fhand = open(**"stuff.jpg"**, **"wb"**)  
fhand.write(picture)  
fhand.close()

***Резултат****: 4236 4236*

*5120 9356*

*3352 12708*

*1772 14480*

*5120 19600*

*5120 24720*

*3988 28708*

*4236 32944*

*5120 38064*

*5120 43184*

*256 43440*

*5120 48560*

*5120 53680*

*5120 58800*

*5120 63920*

*5120 69040*

*5120 74160*

*5120 79280*

*5120 84400*

*5120 89520*

*1488 91008*

*5120 96128*

*5120 101248*

*112 101360*

*5120 106480*

*5120 111600*

*5120 116720*

*5120 121840*

*5120 126960*

*5120 132080*

*5120 137200*

*5120 142320*

*5120 147440*

*5120 152560*

*5120 157680*

*5120 162800*

*5120 167920*

*32 167952*

*1448 169400*

*5120 174520*

*5016 179536*

*1448 180984*

*1412 182396*

*1412 183808*

*5120 188928*

*528 189456*

*5120 194576*

*5120 199696*

*1056 200752*

*5120 205872*

*5120 210992*

*4760 215752*

*5120 220872*

*5120 225992*

*4616 230608*

*Header length 394*

*HTTP/1.1 200 OK*

*Date: Sat, 18 Mar 2023 15:31:41 GMT*

*Server: Apache/2.4.18 (Ubuntu)*

*Last-Modified: Mon, 15 May 2017 12:27:40 GMT*

*ETag: "38342-54f8f2e5b6277"*

*Accept-Ranges: bytes*

*Content-Length: 230210*

*Vary: Accept-Encoding*

*Cache-Control: max-age=0, no-cache, no-store, must-revalidate*

*Pragma: no-cache*

*Expires: Wed, 11 Jan 1984 05:00:00 GMT*

*Connection: close*

*Content-Type: image/jpeg*

Може да се види, че за този URL адрес Content-Type показва, че тялото на документа е изображение (image / jpeg).

Докато се изпълнява програмата, може да се види, че не получаваме 5120 знака всеки път, когато се обърнем към метода recv(). Получаваме толкова символи, колкото са ни прехвърлени през мрежата от уеб сървъра в момента, в който извикаме recv().

Може да се забави последователността чрез използване на time.sleep(). Със закъснението програмата се изпълнява по следния начин:

***Резултат****: 4236 4236*

*5120 9356*

*5120 14476*

*5120 19596*

*5120 24716*

*5120 29836*

*5120 34956*

*5120 40076*

*5120 45196*

*5120 50316*

*5120 55436*

*5120 60556*

*5120 65676*

*5120 70796*

*5120 75916*

*5120 81036*

*5120 86156*

*5120 91276*

*5120 96396*

*5120 101516*

*5120 106636*

*5120 111756*

*5120 116876*

*5120 121996*

*5120 127116*

*5120 132236*

*5120 137356*

*5120 142476*

*5120 147596*

*5120 152716*

*5120 157836*

*5120 162956*

*5120 168076*

*5120 173196*

*5120 178316*

*5120 183436*

*5120 188556*

*5120 193676*

*5120 198796*

*5120 203916*

*5120 209036*

*5120 214156*

*5120 219276*

*5120 224396*

*5120 229516*

*1092 230608*

*Header length 394*

*HTTP/1.1 200 OK*

*Date: Sat, 18 Mar 2023 15:33:37 GMT*

*Server: Apache/2.4.18 (Ubuntu)*

*Last-Modified: Mon, 15 May 2017 12:27:40 GMT*

*ETag: "38342-54f8f2e5b6277"*

*Accept-Ranges: bytes*

*Content-Length: 230210*

*Vary: Accept-Encoding*

*Cache-Control: max-age=0, no-cache, no-store, must-revalidate*

*Pragma: no-cache*

*Expires: Wed, 11 Jan 1984 05:00:00 GMT*

*Connection: close*

*Content-Type: image/jpeg*

По този начин чрез recv(), се получават 5120 знака всеки път, когато се поискат нови данни.

**4. Извличане на уеб страници с urllib**

Въпреки, че може ръчно да се изпращат и получават данни през HTTP чрез библиотеката на сокетите, има много по-опростен начин да се изпълни тази обща задача в Python, използвайки библиотеката urllib.

Използвайки urllib, може уеб страница да се третира подобно на файл. Чрез посочване на уеб страница за изтегляне urllib обработва всички HTTP протоколи и подробности за описанието.

Еквивалентният код за четене на romeo.txt файла от интернет с помощта на urllib е:

import urllib.request

fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')

**for** line in fhand:

print(line.decode().strip())

След като уеб страницата е отворена с urllib.urlopen, може да се третира като файл и да се чете, използвайки for loop.

Когато програмата работи, се вижда само изхода на съдържанието на файла.

**Самостоятелна задача 1:**

Да се напише програма за извличане на данните от файл и да се изчисли честотата на всяка дума във файла.

import socket

import time

import urllib.request

import re

str\_fhand = ""

fhand = urllib.request.urlopen("http://data.pr4e.org/romeo.txt")

for line in fhand:

    str\_fhand += line.decode().strip()

list\_fhand = re.sub("[^\w]", " ", str\_fhand).split()

print(list\_fhand)

for list\_fhand\_iter in list\_fhand:

    print(f"{list\_fhand\_iter} : {list\_fhand.count(list\_fhand\_iter)}")

['But', 'soft', 'what', 'light', 'through', 'yonder', 'window', 'breaksIt', 'is', 'the', 'east', 'and', 'Juliet', 'is', 'the', 'sunArise', 'fair', 'sun', 'and', 'kill', 'the', 'envious', 'moonWho', 'is', 'already', 'sick', 'and', 'pale', 'with', 'grief']

But : 1

soft : 1

what : 1

light : 1

through : 1

yonder : 1

window : 1

breaksIt : 1

is : 3

the : 3

east : 1

and : 3

Juliet : 1

is : 3

the : 3

sunArise : 1

fair : 1

sun : 1

and : 3

kill : 1

the : 3

envious : 1

moonWho : 1

is : 3

already : 1

sick : 1

and : 3

pale : 1

with : 1

grief : 1

**5. Четене на двоични файлове с помощта на urllib.**

Данните в бинарните файлове по принцип не са подходящи за разпечатване, но лесно може да се направи копие на URL в локален файл на твърд диск с помощта на urllib.

Необходимо е да се отвори URL адреса и да се прочете, за да се изтегли цялото съдържание на документа в низ (img), след което да се запише тази информация в локален файл.

**Примерна реализация**

import urllib.request, urllib.parse, urllib.error

img = urllib.request.urlopen('http://data.pr4e.org/cover3.jpg').read()

fhand = open('cover3.jpg', 'wb')

fhand.write(img)

fhand.close()

Тази програма чете всички данни наведнъж в мрежата и ги съхранява в променливата img в оперативната памет, след това отваря файла cover.jpg и записва данните на диска. Аргументът wb на функция open() отваря двоичен файл само за писане. Тази програма ще работи, ако размерът на файла е по-малък от размера на паметта на компютъра.

Ако обаче това е голям аудио или видео файл, тази програма може да се срине или поне да работи изключително бавно, когато на компютъра липсва памет. За да се избегне изчерпване на паметта, се извличат данните в блокове (или буфери) и след това се записват на диска, преди да се извлече следващия блок. По този начин програмата може да чете файл с всякакъв размер, без да използва цялата памет, която има компютърът.

import urllib.request, urllib.parse, urllib.error

img = urllib.request.urlopen('http://data.pr4e.org/cover3.jpg')

fhand = open('cover3.jpg', 'wb')

size = 0

**while** True:

info = img.read(100000)

**if** len(info) < 1: **break**

size = size + len(info)

fhand.write(info)

print(size, 'characters copied.')

fhand.close()

В този пример се четат само 100 000 знака наведнъж и след това се записват във файла cover.jpg, преди да се извлекат следващите 100 000 знака от мрежата.

***Резултат****: 230210 characters copied.*

6. Разбор на HTML с помощта на регулярни изрази

Един прост начин за разбор на HTML е използването на регулярни изрази за многократно търсене и извличане на подстраници, които съответстват на определен модел. Ето една проста уеб страница:

**<h1>**The First Page**</h1>**

**<p>**

If you like, you can switch to the

**<a** href="http://www.dr-chuck.com/page2.htm"**>**

Second Page**</a>**.

**</p>**

Следният регулярен израз търси съвпадения и извлича стойностите на връзката от горния текст:

href="http[s]?://.+?"

**Самостоятелна задача 2:**

Напишете програма, която извлича адреси при съвпадение с въведен регулярен израз.

import socket

import time

import urllib.request, urllib.parse, urllib.error

import re

import requests

from bs4 import BeautifulSoup

import bs4

webpage = str(

    urllib.request.urlopen(

        "https://www.transportation.gov/gis/national-address-database"

    ).read()

)

soup = bs4.BeautifulSoup(webpage)

# print(soup.get\_text())

str\_fhand = soup.get\_text()

print(

    re.findall(

        "\d+[ ](?:[A-Za-z0-9.-]+[ ]?)+(?:Avenue|Lane|Road|Boulevard|Drive|Street|Ave|Dr|Rd|Blvd|Ln|St)\.?",

        str\_fhand,

    )

)

['0 age of data submission by St', '1200 New Jersey Avenue']

**7. Разбор на HTML с помощта на BeautifulSoup**

Има редица библиотеки на Python, които могат да помогнат при анализиране и извличане на данни от HTML страниците. Всяка от библиотеките има своите силни и слаби страни. Библиотеката BeautifulSoup позволява лесно да се извличат необходимите данни. Можете да изтеглите и инсталирате BeautifulSoup от:

*https://pypi.python.org/pypi/beautifulsoup4*

Информацията за инсталиране на BeautifulSoup с инструмента Python Package Index е достъпна на:

*https://packaging.python.org/tutorials/installing-packages/*

***Пример:***

Използване на urllib за четене на страницата и след това чрез BeautifulSoup се извличат атрибутите на href от маркерите anchor (a).

import urllib.request, urllib.parse, urllib.error

from bs4 import BeautifulSoup

import ssl

*# Ignore SSL certificate errors*

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

url = input('Enter - ')

html = urllib.request.urlopen(url, context=ctx).read()

soup = BeautifulSoup(html, 'html.parser')

*# Retrieve all of the anchor tags*

tags = soup('a')

**for** tag in tags:

print(tag.get('href', None))

Програмата подканва за уеб адрес, след това отваря уеб страницата, чете данните и ги предава на парсера BeautifulSoup и след това извлича всички анкерни маркери и отпечатва атрибута href за всеки маркер.

***Резултат****: Enter - https://www.transportation.gov/gis/national-address-database*

*#main*

*/*

*/about*

*/meet-secretary/secretary-pete-buttigieg*

*/about*

*/mission/meet-key-officials*

*/newsroom*

*/dot-strategic-plan*

*/budget*

*/careers*

*https://www.faa.gov/*

*https://highways.dot.gov/*

*https://www.fmcsa.dot.gov/*

*https://railroads.dot.gov/*

*https://www.transit.dot.gov/*

*https://www.seaway.dot.gov/*

*https://www.maritime.dot.gov/*

*https://www.nhtsa.gov/*

*https://www.oig.dot.gov/*

*/office-of-secretary*

*https://www.phmsa.dot.gov/*

*/priorities*

*/bipartisan-infrastructure-law*

*/priorities/climate-sustainability*

*/priorities/equity*

*/priorities/safety*

*/priorities/transformation*

*/contact-us*

*/blog/Destinations-by-DOT*

*/social*

*/contact-us*

*https://www.youtube.com/user/usdotgov*

*https://www.facebook.com/USDOT*

*https://twitter.com/USDOT*

*https://www.instagram.com/usdot/*

*https://www.linkedin.com/company/us-department-of-transportation-dot*

*https://medium.com/destinations-by-dot*

*/*

*/mission/about-us*

*/gis*

*/gis*

*/gis/national-address-database/national-address-database-0*

*/gis/national-address-database/national-address-database-0*

*/gis/nad/nad-schema*

*/gis/national-address-database/national-address-database-nad-disclaimer*

*https://www.fgdc.gov/organization/working-groups-subcommittees/address-sc/index\_html*

*/tags/gis*

*/tags/nad*

*None*

*https://storymaps.arcgis.com/stories/9490f773f65d4c6aa8b79facc528a661*

*mailto:nad@dot.gov*

*#map1*

*#map2*

*#map3*

*None*

*None*

*None*

*https://www.facebook.com/USDOT*

*https://twitter.com/USDOT*

*https://www.instagram.com/usdot/*

*https://www.linkedin.com/company/us-department-of-transportation-dot*

*https://medium.com/destinations-by-dot*

*https://public.govdelivery.com/accounts/usdot/subscribers/new*

*/meet-secretary/secretary-pete-buttigieg*

*/about*

*/newsroom*

*/blog/Destinations-by-DOT*

*/social*

*/mission/meet-key-officials*

*/regulations*

*https://www.transportation.gov/transerve*

*/careers*

*/contact-us*

*https://www.faa.gov/*

*https://highways.dot.gov/*

*https://www.fmcsa.dot.gov/*

*https://railroads.dot.gov/*

*https://www.transit.dot.gov/*

*https://www.seaway.dot.gov/*

*https://www.maritime.dot.gov/*

*https://www.nhtsa.gov/*

*https://www.oig.dot.gov/*

*/office-of-secretary*

*https://www.phmsa.dot.gov/*

*/administrations/research-and-technology*

*https://www.bts.gov/*

*https://www.volpe.dot.gov/*

*https://transportation.libanswers.com/*

*/bipartisan-infrastructure-law*

*/priorities/climate-sustainability*

*/priorities/equity*

*/safety*

*/priorities/transformation*

*https://www.usa.gov/*

*/dot-website-privacy-policy*

*/foia*

*/budget*

*/civil-rights/civil-rights-awareness-enforcement/no-fear-act*

*/mission/civil-rights/elijah-e-cummings-federal-employee-antidiscrimination-act-2020-agency-report*

*/ethics*

*/web-policies*

*/vulnerability-disclosure-policy*

*/accessibility*

**Самостоятелна задача 3:**

Напишете програма за изтегляне на различни части от маркер.

import urllib.request, urllib.parse, urllib.error

from bs4 import BeautifulSoup

import ssl

# Ignore SSL certificate errors

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

url = input("Enter - ")

html = urllib.request.urlopen(url, context=ctx).read()

soup = BeautifulSoup(html, "html.parser")

# Retrieve all of the anchor tags

tags = soup("div")

for tag in tags:

    print(tag.get("id", None))

    print(tag.get("class", None))

['sidecontent']

None

['wrapper\_\_related\_links']

None

['sidebar\_\_content\_block', 'mb-5']

None

['sidebar\_\_content\_block', 'mb-5']

None

['card']

None

['card-body']

None

['wrapper\_\_contact\_info']

content

['main-content', 'col']

block-transpo-content

['block-system', 'block-system-main-block', 'clearfix']

None

['node\_\_content', 'mt-0', 'clearfix']

None

['mb-4', 'clearfix']

None

['w-100', 'clearfix']

None

['mb-4', 'py-4']

None

['site-footer\_\_top', 'clearfix']

None

['container']

None

['row']

None

['col-lg-4', 'mb-4', 'order-first']

block-footeraddress

['block', 'block-fixed-block-content', 'block-fixed-block-contentfooter-address']

None

['content']

None

['clearfix', 'text-formatted', 'field', 'field--name-body', 'field--type-text-with-summary', 'field--label-hidden', 'field\_\_item']

block-footer-nav-social

['block', 'block-fixed-block-content', 'block-fixed-block-contentfooter-social']

None

['content']

None

['clearfix', 'text-formatted', 'field', 'field--name-body', 'field--type-text-with-summary', 'field--label-hidden', 'field\_\_item']

block-footersubscription

['block', 'block-subscription', 'block-fixed-block-content', 'block-fixed-block-contentfooter-subscription', 'clearfix']

None

['content']

None

['clearfix', 'text-formatted', 'field', 'field--name-body', 'field--type-text-with-summary', 'field--label-hidden', 'field\_\_item']

None

['mb-4', 'pr-4']

None

['col-sm-6', 'col-lg-2', 'order-2']

None

['col-sm-6', 'col-lg-2', 'order-3']

None

['col-sm-6', 'col-lg-2', 'order-4']

None

['col-sm-6', 'col-lg-2', 'order-last']

None

['site-footer\_\_bottom', 'clearfix']

None

['container']

**Самостоятелна задача 4:**

Напишете програма за осъществяване на връзка със сървър, така че да може да се зарежда всяка уеб страница. Можете да използвате split ('/') за разбиване на URL адреса на съставните му части, за да можете да извлечете името на хоста за повикване на сокет. Добавете проверка за грешка, при която потребителят въвежда неправилно форматиран или несъществуващ URL адрес.

import urllib.request, urllib.parse, urllib.error

from bs4 import BeautifulSoup

import ssl

from requests import HTTPError

from requests.models import PreparedRequest

import requests.exceptions

def check\_url(url) -> bool:

    prepared\_request = PreparedRequest()

    try:

        prepared\_request.prepare\_url(url, None)

    except requests.exceptions.MissingSchema as e:

        return False

    return True

# Ignore SSL certificate errors

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

url = input("Enter - ")

if check\_url(url):

    page = requests.get(url)

    data = page.text

    soup = BeautifulSoup(data)

    list\_href = []

    for link in soup.find\_all("a"):

        link\_href = str(link.get("href"))

        if link\_href.startswith("/") and len(link\_href) > 1:

            list\_href.append(link\_href)

    print(

        f"All webpages from: {url}\n\n\

          {list\_href}"

    )

    for list\_href\_iter in list\_href:

        print(url + list\_href\_iter)

        try:

            urllib.request.urlopen(url + list\_href\_iter, context=ctx).read()

        except BaseException as err:

            print(f"\nError for: {url + list\_href\_iter}")

            pass

# https://www.transportation.gov/gis/national-address-database

else:

    print("ERROR! Url is not valid!")

All webpages from: https://aws.amazon.com

['/contact-us/?nc2=h\_header', '/products/?nc2=h\_ql\_prod', '/solutions/?nc2=h\_ql\_sol', '/pricing/?nc2=h\_ql\_pr', '/getting-started/?nc2=h\_ql\_le', '/partners/?nc2=h\_ql\_pn', '/customer-enablement/?nc2=h\_ql\_ce', '/events/?nc2=h\_ql\_ev', '/contact-us/?nc2=h\_ql\_exm', '/profile/?nc2=h\_m\_mc', '/premiumsupport/knowledge-center/?nc2=h\_m\_ma', '/premiumsupport/?nc2=h\_m\_bc', '/products/?nc2=h\_mo', '/solutions/?nc2=h\_mo', '/pricing/?nc2=h\_mo', '/what-is-aws/?nc2=h\_mo', '/getting-started/?nc2=h\_mo', '/training/?nc2=h\_mo', '/developer/?nc2=h\_mo', '/solutions/case-studies/?nc2=h\_mo', '/partners/?nc2=h\_mo', '/console/mobile/', '/what-is-aws/?sc\_icampaign=aware\_what\_is\_aws\_default&sc\_ichannel=ha&sc\_icontent=awssm-evergreen\_pac\_default&sc\_iplace=hero&trk=ha\_awssm-evergreen\_pac\_default', '/redshift/?sc\_icampaign=pac\_gc-600-alt\_redshift\_pdp\_w32y22&sc\_ichannel=ha&sc\_icontent=awssm-11187\_pac\_customer&sc\_iplace=hero&trk=eeceb344-103a-4a17-bfab-1dc029f7a432~ha\_awssm-11187\_pac\_customer', '/s3/features/object-lambda/?sc\_icampaign=aware\_s3\_object\_lambda\_w35y22&sc\_ichannel=ha&sc\_icontent=awssm-11209\_pac\_customer&sc\_iplace=hero&trk=39a7b81b-f182-429e-aa7c-c6698b81697c~ha\_awssm-11209\_pac\_customer', '/training/digital/?sc\_icampaign=aware\_digitaltraining\_skillbuilder\_subscription\_hero\_aug\_tnc\_global\_traincert\_100-dl&sc\_ichannel=ha&sc\_icontent=awssm-11189\_tnc\_customer&sc\_iplace=hero&trk=7cecd3f6-df6d-48e5-acfb-6e9f1a23c2a8~ha\_awssm-11189\_tnc~ha\_awssm-11189\_tnc\_customer', '/free/?sc\_icampaign=acq\_freetier-default&sc\_ichannel=ha&sc\_icontent=awssm-evergreen-default&sc\_iplace=ed&trk=ha\_awssm-evergreen-default', '/getting-started/?sc\_icampaign=grizzly\_getting-started-center\_default-ed&sc\_ichannel=ha&sc\_icontent=awssm-evergreen-default&sc\_iplace=ed&trk=ha\_awssm-evergreen-default', '/remote-work-learning/?sc\_icampaign=launch\_4028\_remote-work\_June30\_default-ed&sc\_ichannel=ha&sc\_icontent=awssm-4028\_default&sc\_iplace=ed&trk=ha\_awssm-4028\_default', '/free/compute/lightsail/?sc\_icampaign=Adoption\_Campaign\_lightsail\_default-ed\_20202707&sc\_ichannel=ha&sc\_icontent=awssm-4660\_default-ed&sc\_ioutcome=Enterprise\_Digital\_Marketing&sc\_iplace=ed&trk=ha\_awssm-4660\_default-ed&trkCampaign=pac-edm-2020-lightsail-free-page-test', '/advertising-marketing/?hp=tile&tile=industries', '/financial-services/?hp=tile&tile=industries', '/gametech/?hp=tile&tile=industries', '/stateandlocal/?hp=tile&tile=industries', '/industries/?hp=tile&tile=industries', '/big-data/datalakes-and-analytics/?hp=tile&tile=solutions', '/machine-learning/?hp=tile&tile=solutions', '/serverless/?hp=tile&tile=solutions', '/products/storage/?hp=tile&tile=solutions', '/solutions/?hp=tile&tile=solutions', '/products/?hp=tile&so-exp=below', '/big-data/datalakes-and-analytics/?hp=tile&so-exp=below/#AWS\_Analytics\_services', '/aws-cost-management/?hp=tile&so-exp=below/#AWS\_Cost\_Management\_Services', '/products/compute/?hp=tile&so-exp=below/#AWS\_Compute\_services', '/containers/?hp=tile&so-exp=below/#AWS\_Containers\_services', '/products/databases/?hp=tile&so-exp=below/#Database\_services', '/products/frontend-web-mobile/?hp=tile&so-exp=below/#AWS\_Front-End\_Web\_.26\_Mobile\_tools\_.26\_services', '/iot/?hp=tile&so-exp=below/#AWS\_IoT\_services', '/machine-learning/?hp=tile&so-exp=below/#Explore\_AWS\_Machine\_Learning\_services', '/products/networking/?hp=tile&so-exp=below/#AWS\_Networking\_and\_Content\_Delivery\_services', '/products/security/?hp=tile&so-exp=below/#AWS\_Security.2C\_Identity.2C\_.26\_Compliance\_services', '/serverless/?hp=tile&so-exp=below/#Serverless\_Services\_on\_AWS', '/products/storage/?hp=tile&so-exp=below', '/training/ramp-up-guides/?hp=tile&tile=crt', '/certification/?hp=tile&tile=crt', '/solutions/case-studies/?hp=tile&tile=customerstories', '/security/?hp=tile&tile=security', '/compliance/?hp=tile&tile=security', '/hybrid/?hp=tile&tile=security', '/pricing/?hp=tile&tile=security', '/about-aws/global-infrastructure/?hp=tile&tile=map', '/resources/analyst-reports/?sc\_icampaign=aware\_analyst\_reports&sc\_ichannel=ha&sc\_icontent=awssm-evergreen-analyst\_reports-default-editorial-lower&sc\_iplace=ed&trk=ha\_awssm-evergreen-analyst\_reports-default-editorial-lower', '/training/?sc\_icampaign=aware\_aws-training&sc\_ichannel=ha&sc\_icontent=awssm-1749-default-editorial-lower&sc\_iplace=ed&trk=ha\_awssm-1749-default-editorial-lower', '/partners/?sc\_icampaign=aware\_apn\_recruit&sc\_ichannel=ha&sc\_icontent=awssm-evergreen-apn\_recruit-default-editorial-lower&sc\_iplace=ed&trk=ha\_awssm-evergreen-apn\_recruit-default-editorial-lower', '/builders-library/?cards-body.sort-by=item.additionalFields.customSort&cards-body.sort-order=asc&sc\_icampaign=aware\_builders-library&sc\_ichannel=ha&sc\_icontent=awssm-6762\_aware&sc\_iplace=ed&trk=ha\_awssm-6762\_aware', '/what-is-aws/?nc1=f\_cc', '/what-is-cloud-computing/?nc1=f\_cc', '/diversity-inclusion/?nc1=f\_cc', '/devops/what-is-devops/?nc1=f\_cc', '/containers/?nc1=f\_cc', '/big-data/datalakes-and-analytics/what-is-a-data-lake/?nc1=f\_cc', '/security/?nc1=f\_cc', '/new/?nc1=f\_cc', '/blogs/?nc1=f\_cc', '/getting-started/?nc1=f\_cc', '/training/?nc1=f\_cc', '/solutions/?nc1=f\_cc', '/architecture/?nc1=f\_cc', '/faqs/?nc1=f\_dr', '/resources/analyst-reports/?nc1=f\_cc', '/partners/work-with-partners/?nc1=f\_dr', '/developer/?nc1=f\_dr', '/developer/tools/?nc1=f\_dr', '/developer/language/net/?nc1=f\_dr', '/developer/language/python/?nc1=f\_dr', '/developer/language/java/?nc1=f\_dr', '/developer/language/php/?nc1=f\_cc', '/developer/language/javascript/?nc1=f\_dr', '/contact-us/?nc1=f\_m', '/premiumsupport/knowledge-center/?nc1=f\_dr', '/premiumsupport/?nc1=f\_dr', '/legal/?nc1=f\_cc', '/podcasts/']

https://aws.amazon.com/contact-us/?nc2=h\_header

Page loaded!: https://aws.amazon.com/contact-us/?nc2=h\_header

https://aws.amazon.com/products/?nc2=h\_ql\_prod

Page loaded!: https://aws.amazon.com/products/?nc2=h\_ql\_prod

https://aws.amazon.com/solutions/?nc2=h\_ql\_sol

Page loaded!: https://aws.amazon.com/solutions/?nc2=h\_ql\_sol

https://aws.amazon.com/pricing/?nc2=h\_ql\_pr

Page loaded!: https://aws.amazon.com/pricing/?nc2=h\_ql\_pr

https://aws.amazon.com/getting-started/?nc2=h\_ql\_le

Page loaded!: https://aws.amazon.com/getting-started/?nc2=h\_ql\_le

https://aws.amazon.com/partners/?nc2=h\_ql\_pn

Page loaded!: https://aws.amazon.com/partners/?nc2=h\_ql\_pn

https://aws.amazon.com/customer-enablement/?nc2=h\_ql\_ce

Page loaded!: https://aws.amazon.com/customer-enablement/?nc2=h\_ql\_ce

https://aws.amazon.com/events/?nc2=h\_ql\_ev

Page loaded!: https://aws.amazon.com/events/?nc2=h\_ql\_ev

https://aws.amazon.com/contact-us/?nc2=h\_ql\_exm

**Самостоятелна задача 5:**

Променете вашата програма от самостоятелна задача 4, така че да се преброи броя на получените символи и да спре да се показва текст, след като са показани 300 символа. Програмата трябва да извлече целия документ и да преброи общия брой символи и да покаже броя на символи в края на документа. Тествайте например с url 'http://data.pr4e.org/romeo.txt'.

**Самостоятелна задача 6:**

Използвайте urllib, за да реализирате задача 4 и задача 5.

**Самостоятелна задача 7:**

Напишете програма за извличане, преброяване и извеждане на параграфите в HTML документ.

**Самостоятелна задача 8:**

Променете сокетната програма от самостоятелна задача 5, така че тя да показва данни само след като са получени хедърите и празният ред. Не забравяйте, че recv() получава символи (включително и нови редове).

**Полезни връзки.**

* 1. <https://docs.python.org/3/>
  2. <https://www.programiz.com/python-programming>
  3. <https://www.tutorialspoint.com/python/index.htm>
  4. <https://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/ch1.html>

**Допълнителни задачи:**

1. Напишете програма на Python, която проверява дали дадена web страница е налична на сървъра.
2. Напишете програма на Python, която извлича съдържанието на тага title от даден html документ.
3. Напишете програма на Python, която извлича съдържанието на тага h1 от example.com.
4. Напишете програма на Python, която извежда адреса на атрибута **href** на първия таг **a** в даден HTML документ.
5. Напишете програма на Python, която извлича и извежда връзките на всички изображения от en.wikipedia.org/wiki/Peter\_Jeffrey\_(RAAF\_officer).
6. Напишете програма на Python, която проверява дали дадена web страница съдържа заглавие (т.е. title).
7. Напишете програма на Python, която извлича целия текст на дадена web страница.
8. Напишете програма на Python, която извежда всички тагове, които са вложени в тага body на даден html документ (т.е. всички descendants на body).
9. Напишете програма на Python, която извежда всички редове от страницата www.python.org, които съдържат низа “Python”.
10. Напишете програма на Python, която извежда данни за 250 най-гледани филми от IMDB (име на филм, дата на премиерата, имена на режисьор и най-известните актьори).