

Appendix 1: Source code urlreader

```
#!/usr/bin/python2.3 -d

#imports
import string
import urllib
import urllib2

def readWebsite(url):
    graph = {}
    global starturl

    url = handleStartUrl(url)

    starturl = url
    getWebpage(url,graph)

    return graph

# Rread info from the webpage
def getWebpage(url, graph):
    url = checkHttp(url)
    try:
        webpage = urllib.urlopen(url)
    except Exception, e:
        print '%s: %s\n' % (e, url)
        return

    content = webpage.read()
    listLinks(content, url, graph)

# remove any indexfile given in the begining
def handleStartUrl(url):
    url = checkHttp(url)
    url = url.lower()
    filextentions = [ ".htm", ".html", ".php", ".shtml", ".py", ".cgi", ".asp", ".jsp" ]
    s = "index"
    for ext in filextentions:
        s2 = s + ext
        if url.rfind(s2) == (len(url)-len(s2)):
            url = url.replace(s2,"")
    if url.rfind("/") != len(url) - 1:
        url = url + "/"

    return url

# Add http:// if this is missing in the url
def checkHttp(url):
    if url.find("http://")== -1:
        url = "http://" + url

    return url

# remove / in the beginning and add the starturl infront of the url
def fixLink(url):
    #remove the / from the beginning of url
    if url.find("/") == 0 and starturl.rfind("/") == len(starturl) -1:
        url = starturl + url[1:]
    #add / on the end of url and add url onto starturl
    elif url.find("/") != 0 and starturl.rfind("/") != len(starturl) -1:
        url = starturl + "/" + url
    #add url onto starturl
```

```

else:
    url = starturl + url

return url

# scan the webpage and retrieve a list of strings with hyperlinks
def listLinks(content, url, graph):
    parser = LinkParser()
    try:
        parser.feed(content)
    except Exception, e:
        print '%s: %s\n' % (e, url)
        return
    parser.close()
    hrefs = {}
    i = 0
    for href in parser.hrefs:
        hrefs[i] = href
        i = i + 1
    classifyLink(hrefs, url, graph)

# find all the links from the file
from HTMLParser import HTMLParser
class LinkParser(HTMLParser):
    def reset(self):
        HTMLParser.reset(self)
        self.hrefs = []
    def handle_starttag(self, tag, attrs):
        href = [v for k,v in attrs if k.lower() == 'href']
        if href: self.hrefs.extend(href)

# check if the is internal or external or not a link to any page at all
def classifyLink(hrefs, url, graph):
    i = 0
    internals = []
    externals = []
    notpagelinks = []
    mails = []
    # fileextentions which are not regular links
    fileextentions =
[".css", ".jvs", ".zip", ".exe", ".rar", ".tar", ".py", ".c", ".cpp", ".h", ".ppd", ".tgz", ".gz", ".s
it", ".dmg", ".rpm", ".dep", ".mdb", ".sql", ".bat", ".pls", ".raw", ".pdf", ".doc", ".xls", ".ppt", ".
pps", ".jpg", ".jpeg", ".gif", ".png", ".ico", ".bmp", ".wmv", ".mpg", ".mpeg", ".avi", ".ram", ".mo
v", ".wav", ".mp3", ".mwa"]

#classfy the links
while i < len(hrefs.values()):
    # reset variables
    k = 0
    notvalid = 0

    # check each of the file extentions with the end of the link
    while k < len(fileextentions):
        if (hrefs[i].rfind(fileextentions[k]) == (len(hrefs[i]) -
len(fileextentions[k]))) and hrefs[i].rfind(fileextentions[k]) != -1:
            notvalid = 1
            k = k + 1

    #if the link was not valid
    if notvalid == 1:
        notpagelinks.append(hrefs[i])

    # if the link contains the url and is valid
    elif hrefs[i].find(starturl) != -1:
        internals.append(hrefs[i])

    # if the link contains http:// and is valid and do not contain the url
    elif hrefs[i].find("http://") == 0:

```

```

        externals.append(hrefs[i])

    elif hrefs[i].find("mailto:") == 0:
        mails.append(hrefs[i])

    # the link is valid and do not contain the url or http:// or mail
    else:
        internals.append(fixLink(hrefs[i]))

    i = i + 1
# Sett all the interal links into the graph
graph[url]= internals

# Traverse the webpage
traverseWebpage(url, graph)

# recursive function which travers the webpage
def traverseWebpage(url, graph):
    numboflinks = len(graph[url])
    i = 0
    # traverse the link
    while i < numboflinks:
        #traverse every page wich has not ben traversed befor
        if not (graph.has_key(graph[url][i])):
            getWebpage(graph[url][i],graph)
            i = i + 1
    #save the graph to a global variable
    setGraph(graph)

# set value to a global variable
def setGraph(graph):
    global sitegraph
    sitegraph = graph

```