```
"""
https://code.google.com/p/imakerobots/wiki/netcrawler

Copyright (c) 2001, Tony Mendoza
All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

Redistributions of source code must retain the above
copyright notice, this list of conditions and the
following disclaimer.

Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the
following disclaimer in the documentation and/or other
materials provided with the distribution.

Neither the name of Tony Mendoza nor the names of
its contributors may be used to endorse or promote
products derived from this software without specific
prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

"""

import htmllib
import httplib
import urllib
import formatter
import sys, os, os.path
import urlparse
import string, re
import socket

class NCImage:

        def __init__(self, filename, height=0, width=0, alt=""):
```

```
        self.filename = filename
        self.height = height
        self.width = width
        self.alt = alt
        self.text = ""
        self.isAbsolute = 0
        self.isRoot = 0

        if re.match("^[Hh][Tt][Tt][Pp]+.*", self.filename) != None:
            self.isAbsolute = 1

        if re.match("^/+.*", self.filename) != None:
            self.isRoot = 1

    def get_filename(self):
        return self.filename

    def set_filename(self, filename):
        self.filename = filename
        if re.match("^[Hh][Tt][Tt][Pp]+.*", self.filename) != None:
            self.isAbsolute = 1

        if re.match("^/+.*", self.filename) != None:
            self.isRoot = 1

    def get_height(self):
        return self.height

    def set_height(self, height):
        self.height = height

    def get_width(self):
        return self.width

    def set_width(self, width):
        self.width = width

    def get_alt(self):
        return self.alt

    def set_alt(self, alt):
        self.alt = alt

    def get_isAbsolute(self):
        return self.isAbsolute

    def get_isRoot(self):
        return self.isRoot

    def get_text(self):
        return self.text

    def set_text(self, text):
```

```
                self.text = text

        def __str__(self):
                message = "\t<Image>\n"
                message = message + "\t\t<Filename>%s</Filename>\n" % self.filename
                message = message + "\t\t<Width>%s</Width>\n" % self.width
                message = message + "\t\t<Height>%s</Height>\n" % self.height
                message = message + "\t\t<Alt>%s</Alt>\n" % self.alt
                message = message + "\t</Image>\n"
                return message

        def get_xml(self):
                message = "\t<Image>\n"
                message = message + "\t\t<Filename>%s</Filename>\n" % self.filename
                message = message + "\t\t<Width>%s</Width>\n" % self.width
                message = message + "\t\t<Height>%s</Height>\n" % self.height
                message = message + "\t\t<Alt>%s</Alt>\n" % self.alt
                message = message + "\t</Image>\n"
                return message


class NCMeta:

        def __init__(self):

                self.meta_type = ""
                self.content = ""

        def get_name(self):
                return self.meta_type

        def set_name(self, data):
                self.meta_type = data

        def get_content(self):
                return self.content

        def set_content(self,content):
                self.content = content

        def __str__(self):
                message = "\t<Meta>\n"
                message = message + "\t\t<Type>%s</Type>\n" % self.meta_type
                message = message + "\t\t<Content>%s</Content>\n" % self.content
                message = message + "\t</Meta>\n"
                return message

        def get_xml(self):
                message = "\t<Meta>\n"
                message = message + "\t\t<Type>%s</Type>\n" % self.meta_type
                message = message + "\t\t<Content>%s</Content>\n" % self.content
                message = message + "\t</Meta>\n"
                return message
```

```python
class NCFrame:

        def __init__(self):

                self.src = ""
                self.name = ""

        def get_name(self):
                return self.name

        def get_src(self):
                return self.src

        def set_src(self, src):
                self.src = src

        def set_name(self, name):
                self.name = name

        def __str__(self):
                message = "\t<Frame>\n"
                if self.src != "":
                        message = message + "\t\t<Src>%s</Src>\n" % self.src
                if self.name != "":
                        message = message + "\t\t<Name>%s</Name>\n" % self.name
                message = message + "\t</Frame>\n"
                return message

        def get_xml(self):
                message = "\t<Frame>\n"
                if self.src != "":
                        message = message + "\t\t<Src>%s</Src>\n" % self.src
                if self.name != "":
                        message = message + "\t\t<Name>%s</Name>\n" % self.name
                message = message + "\t</Frame>\n"
                return message

class NCFrameset:

        def __init__(self):

                self.rows = 0
                self.columns = 0
                self.frames = []

        def get_rows(self):
                return self.rows

        def get_cols(self):
                return self.columns

        def get_frames(self):
                return self.frames

        def set_cols(self, cols):
                self.columns = cols

        def set_rows(self, rows):
                self.rows = rows

        def add_frame(self, frame):
                self.frames.append(frame)

        def __str__(self):
                message = "\t<Frameset>\n"
                if self.rows > 0:
                        message = message + "\t\t<Rows>%s</Rows>\n" % self.rows
                if self.columns > 0:
                        message = message + "\t\t<Cols>%s</Cols>\n" % self.columns
                if len(self.frames) > 0:
                        for x in self.frames:
                                message = message + "\t\t<Frame>\n"
                                if x.src != "":
                                        message = message + "\t\t\t<Src>%s</Src>\n" % x.src
                                if x.name != "":
                                        message = message + "\t\t\t<Name>%s</Name>\n" % x.name
                                message = message + "\t\t</Frame>\n"
                message = message + "\t</Frameset>\n"
                return message

        def get_xml(self):
                message = "\t<Frameset>\n"
                if self.rows > 0:
                        message = message + "\t\t<Rows>%s</Rows>\n" % self.rows
                if self.columns > 0:
                        message = message + "\t\t<Cols>%s</Cols>\n" % self.columns
                if len(self.frames) > 0:
                        for x in self.frames:
                                message = message + "\t\t<Frame>\n"
                                if x.src != "":
                                        message = message + "\t\t\t<Src>%s</Src>\n" % x.src
                                if x.name != "":
                                        message = message + "\t\t\t<Name>%s</Name>\n" % x.name
                                message = message + "\t\t</Frame>\n"
                message = message + "\t</Frameset>\n"
                return message

class NCAnchor:

        def __init__(self, filename):

                self.filename = filename
                self.text = ""
                self.image = None
                self.isAbsolute = 0
                self.isRoot = 0
                self.parsable = 1

                if re.match("^[Hh][Tt][Tt][Pp]+.*", self.filename) != None:
                        self.isAbsolute = 1

                if re.match("^/+.*$", self.filename) != None:
                        self.isRoot = 1
                else:
                        self.isRelative = 1

                if re.match("^.*[Hh][Tt][Mm]+[Ll]?$|^.*[/]?$", self.filename) == None:
                        self.parsable == 0


        def get_filename(self):
                return self.filename

        def set_filename(self, filename):
                self.filename = filename
                if re.match("^[Hh][Tt][Tt][Pp]+.*", self.filename) != None:
                        self.isAbsolute = 1

                if re.match("^/+.*", self.filename) != None:
                        self.isRoot = 1

        def get_image(self):
                return self.image

        def set_image(self, image):
                self.image = image

        def get_text(self):
                return self.text

        def set_text(self, text):
                self.text = text

        def get_isAbsolute(self):
                return self.isAbsolute

        def get_isRoot(self):
                return self.isRoot

        def __str__(self):
                message = "\t<Anchor>\n"
                message = message +"\t\t<Filename>%s</Filename>\n" % self.filename
                message = message + "\t\t<Text>%s</Text>\n" % self.text

                if self.image != None:
                        message = message + "\t\t<Image>\n"
                        message = message + "\t\t\t<Filename>%s</Filename>\n" % self.image.
                        filename
                        message = message + "\t\t\t<Width>%s</Width>\n" % self.image.width
                        message = message + "\t\t\t<Height>%s</Height>\n" % self.image.height
                        message = message + "\t\t\t<Alt>%s</Alt>\n" % self.image.alt
                        message = message + "\t\t</Image>\n"

                message = message + "\t</Anchor>\n"
                return message

        def get_xml(self):
                message = "\t<Anchor>\n"
                message = message +"\t\t<Filename>%s</Filename>\n" % self.filename
                message = message + "\t\t<Text>%s</Text>\n" % self.text

                if self.image != None:
                        message = message + "\t\t<Image>\n"
                        message = message + "\t\t\t<Filename>%s</Filename>\n" % self.image.
                        filename
                        message = message + "\t\t\t<Width>%s</Width>\n" % self.image.width
                        message = message + "\t\t\t<Height>%s</Height>\n" % self.image.height
                        message = message + "\t\t\t<Alt>%s</Alt>\n" % self.image.alt
                        message = message + "\t\t</Image>\n"

                message = message + "\t</Anchor>\n"
                return message


class NCUrlParser:

        def __init__(self, urlstring):

                self.urlstring = urlstring
                self.parsed_url = urlparse.urlparse(self.urlstring)
                self.protocol = self.parsed_url[0]
                self.split_string = string.split(self.parsed_url[1], ':')
                self.hostname = self.split_string[0]

                if len(self.split_string) == 1:
                        self.port = 80
                else:
                        self.port = self.split_string[1]

                if self.parsed_url[2] == "":
                        self.path = "/"
                else:
                        self.path = self.parsed_url[2]

                self.query = self.parsed_url[3]
                self.query_parameters = self.parsed_url[4]
                self.fragment = self.parsed_url[5]

        def get_url_string(self):
                return self.urlstring
        def get_protocol(self):
                return self.protocol

        def get_hostname(self):
                return self.hostname

        def get_port(self):
                return self.port

        def get_path(self):
                return self.path

        def get_query(self):
                return self.query

        def get_query_param(self):
                return self.query_parameters

        def get_fragment(self):
                return self.fragment

        def set_protocol(self, protocol):
                self.protocol = protocol

        def set_hostname(self, hostname):
                self.hostname = hostname

        def set_port(self, port):
                self.port = port

        def set_path(self, path):
                self.path = path

        def set_query(self, query):
                self.query = query

        def set_query_param(self, query_parameters):
                self.query_parameters = query_parameters

        def set_fragment(self, fragment):
                self.fragment = fragment

class NCError:
        def __init__(self, value):
            self.value = value
        def __str__(self):
            return `self.value`



class NCHTTPInet:
        def __init__(self, urlstring):

                self.url_string = urlstring
                self.url_parser = NCUrlParser(urlstring)
                self.http_conn = None
                self.error_code = ""
                self.error_message = ""
                self.header_list = []
                self.input_stream = None

        def get_input_stream(self):
                if self.input_stream == None:
                        return None
                else:
                        return self.input_stream

        def connect(self):
                try:
                        self.http_conn = httplib.HTTP(self.url_parser.get_hostname(), self.
                        url_parser.get_port())
                        self.http_conn.putrequest('GET', self.url_parser.get_path())
                        self.http_conn.endheaders()
                        self.error_code, self.error_message, self.header_list = self.http_conn.
                        getreply()

                except socket.error, msg:
                        raise NCError(msg.args[0])

                if self.error_code < 300:
                        self.input_stream = self.http_conn.getfile()

        def get_error_code(self):
                return self.error_code

        def get_error_message(self):
                return self.error_message

        def get_response_headers(self):
                return self.header_list

class NCWebDocument(htmllib.HTMLParser):

        def __init__(self, input_stream):
                self.formatter = formatter.NullFormatter()
                htmllib.HTMLParser.__init__(self, self.formatter)
                self.input_stream = input_stream
                self.in_center = 0
                self.in_href = 0
                self.in_head = 0
                self.in_image = 0
                self.in_span = 0
                self.in_caption = 0
                self.in_para = 0
                self.in_table = 0
                self.in_th = 0
                self.in_tr = 0
                self.in_td = 0
                self.in_body = 0
                self.in_title = 0
                self.in_script = 0;
                self.in_frame = 0;
                self.in_frameset = 0;
                self.current_href = None
                self.current_image = None
                self.current_text = ""
                self.current_meta_tag = None
                self.current_frameset_tag = None
                self.current_frame_tag = None
                self.centered_text = []
                self.table_text = []
                self.paragraph_text = []
                self.span_text = []
                self.caption_text = []
                self.body_text = []
                self.tr_text = []
                self.td_text = []
                self.th_text = []
                self.absolute_urls_no_image = []
                self.relative_urls_no_image = []
                self.absolute_urls = []
                self.relative_urls = []
                self.images = []
                self.meta_tags = []
                self.script_tags = []
                self.frameset_tags = []
                self.frame_tags = []
                self.plain_data = []

                self.title_text = ""

        def load(self):
                if self.input_stream != None:
                        self.feed(self.input_stream.read())
                else:
                        return None
                return 1

        def show_anchors(self):
                for x in self.anchorlist:
                        print x

        def get_absolute_urls_no_image(self):
                return self.absolute_urls_no_image

        def get_relative_urls_no_image(self):
                return self.relative_urls_no_image
```

```
	def get_absolute_urls(self):
		return self.absolute_urls

	def get_relative_urls(self):
		return self.relative_urls

	def get_images(self):
		return self.images

	def get_meta_tags(self):
		return self.meta_tags

	######################################
	# ANCHOR TAG

	def start_a(self, attr):
		self.in_href = 1

		for x in attr:
			if re.match("[Hh][Rr][Ee][Ff]", string.strip(x[0])):
				self.current_href = NCAnchor(x[1])

	def end_a(self):
		self.in_href = 0
		if self.current_href != None:
			if self.current_href.get_isAbsolute() == 1:
				if self.current_href.get_image() != None:
					self.absolute_urls.append(self.current_href)
				else:
					self.absolute_urls_no_image.append(self.current_href)

			if self.current_href.get_isAbsolute() == 0:
				if self.current_href.get_image() != None:
					self.relative_urls.append(self.current_href)
				else:
					self.relative_urls_no_image.append(self.current_href)

	# ANCHOR TAG
	######################################

	######################################
	# IMAGE TAG

	def start_img(self, attr):

		self.in_image = 1
		self.current_image = NCImage("nothing")

		for x in attr:
			if re.match("[Ss][Rr][Cc]", string.strip(x[0])):
				self.current_image.set_filename(string.strip(x[1]))
			if re.match("[Ww][Ii][Dd][Tt][Hh]", string.strip(x[0])):
```

```
                                self.current_image.set_width(string.strip(x[1]))
                        if re.match("[Hh][Ee][Ii][Gg][Hh][Tt]", string.strip(x[0])):
                                self.current_image.set_height(string.strip(x[1]))
                        if re.match("[Aa][Ll][Tt]", string.strip(x[0])):
                                self.current_image.set_alt(string.strip(x[1]))

        def end_img(self):
                if self.in_href == 1:
                        self.current_href.set_image(self.current_image)
                else:
                        self.images.append(self.current_image)

                self.in_image = 0

        # IMAGE TAG
        #######################################

        #######################################
        # HEAD TAG

        def start_head(self, attr):
                self.in_head = 1

        def end_head(self):
                self.in_head = 0

        # HEAD TAG
        #######################################

        #######################################
        # META TAG

        def start_meta(self, attr):
                self.current_meta_tag = NCMeta()
                attributes = attr[0]
                attributes2 = attr[1]

                self.current_meta_tag.set_name(attributes[1])
                self.current_meta_tag.set_content(attributes2[1])

                self.meta_tags.append(self.current_meta_tag)

        def end_meta(self):
                pass

        # HEAD TAG
        #######################################

        #######################################
        # DATA HANDLING

        def handle_data(self, data):
```

```
		if string.strip(data) == "":
			return

		if self.in_href:
			self.current_href.set_text(string.strip(data))
		elif self.in_image:
			self.current_image.set_text(string.strip(data))
		elif self.in_center:
			self.centered_text.append(string.strip(data))
		elif self.in_caption:
			self.caption_text.append(string.strip(data))
		elif self.in_para:
			self.paragraph_text.append(string.strip(data))
		elif self.in_span:
			self.span_text.append(string.strip(data))
		elif self.in_table:
			self.table_text.append(string.strip(data))
		elif self.in_th:
			self.th_text.append(string.strip(data))
		elif self.in_tr:
			self.tr_text.append(string.strip(data))
		elif self.in_td:
			self.td_text.append(string.strip(data))
		elif self.in_body:
			self.body_text.append(string.strip(data))
		elif self.in_title:
			self.title_text = string.strip(data)
		elif self.in_script:
			print string.strip(data)
			self.script_tags.append(string.strip(data))
		else:
			self.plain_data.append(string.strip(data))

		return

	# DATA HANDLING
	########################################

	########################################
	# SPAN TAG

	def start_span(self, attr):
		self.in_span = 1

	def end_span(self):
		self.in_span = 0

	# SPAN TAG
	########################################

	########################################
	# PARAGRAPH TAG
```

```
    def start_p(self, attr):
        self.in_para = 1

    def end_p(self):
        self.in_para = 0

    # PARAGRAPH TAG
    #######################################

    #######################################
    # CAPTION TAG

    def start_caption(self, attr):
        self.in_caption = 1

    def end_caption(self):
        self.in_caption = 0

    # CAPTION TAG
    #######################################

    #######################################
    # CENTER TAG

    def start_center(self, attr):
        self.in_center = 1

    def end_center(self):
        self.in_center = 0

    # CENTER TAG
    #######################################

    #######################################
    # TABLE TAG

    def start_table(self, attr):
        self.in_table = 1

    def end_table(self):
        self.in_table = 0

    # TABLE TAG
    #######################################

    #######################################
    # TD TAG

    def start_td(self, attr):
        self.in_td = 1

    def end_td(self):
        self.in_td = 0
    # TD TAG
    ######################################

    ######################################
    # TR TAG

    def start_tr(self, attr):
        self.in_tr = 1

    def end_tr(self):
        self.in_tr = 0

    # TR TAG
    ######################################

    ######################################
    # TH TAG

    def start_th(self, attr):
        self.in_th = 1

    def end_th(self):
        self.in_th = 0

    # TH TAG
    ######################################

    ######################################
    # BODY TAG

    def start_body(self, attr):
        self.in_body = 1

    def end_body(self):
        self.in_body = 0

    # BODY TAG
    ######################################

    ######################################
    # TITLE TAG

    def start_title(self, attr):
        self.in_title = 1

    def end_title(self):
        self.in_title = 0

    # TITLE TAG
    ######################################

    ######################################
```

```
        # SCRIPT TAG

        def start_script(self, attr):
                self.in_script = 1

        def end_script(self):
                self.in_script = 0

        # SCRIPT TAG
        ######################################

        ######################################
        # COMMENT TAG

        def handle_comment(self, comment):
                if self.in_script == 1:
                        for x in string.split(comment, "\n"):
                                obj = re.match(".*([Hh][Tt][Tt][Pp]://[^\'\"]+)+(.)*",x)
                                if obj != None:
                                        self.script_tags.append(obj.group(1))

        # COMMENT TAG
        ######################################

        ######################################
        # FRAMESET TAG

        def start_frameset(self, attr):
                self.in_frameset = 1
                self.current_frameset_tag = NCFrameset()
                for x in attr:
                        if re.match("[Rr][Oo][Ww][Ss]", string.strip(x[0])):
                                self.current_frameset_tag.set_rows(string.strip(x[1]))
                        if re.match("[Cc][Oo][Ll][Ss]", string.strip(x[0])):
                                self.current_frameset_tag.set_cols(string.strip(x[1]))

        def end_frameset(self):
                self.frameset_tags.append(self.current_frameset_tag)
                self.in_frameset = 0

        # FRAMESET TAG
        ######################################

        ######################################
        # FRAME TAG

        def start_frame(self, attr):
                self.in_frame = 1
                self.current_frame_tag = NCFrame()
                for x in attr:
                        if re.match("[Ss][Rr][Cc]", string.strip(x[0])):
                                self.current_frame_tag.set_src(string.strip(x[1]))
                        if re.match("[Nn][Aa][Mm][Ee]", string.strip(x[0])):
                                self.current_frame_tag.set_name(string.strip(x[1]))

        def end_frame(self):
                if self.in_frameset == 1:
                        if self.current_frameset_tag != None:
                                self.current_frameset_tag.frames.append(self.current_frame_tag)
                else:
                        self.frame_tags.append(self.current_frame_tag)
                self.in_frame = 0

        # FRAME TAG
        ######################################

class NetCrawler:

        def __init__(self, outputDir = ""):
                self.headers = None
                self.redirectCount = 0;
                self.url_cache = []
                self.relative_urls = []
                self.ftp_urls = []
                self.weird_urls = []
                self.new_urls = []
                self.new_domains = []
                self.new_url_cache = []
                self.bad_urls = [
                                        'http://dir.yahoo.com',
                                        'http://www.yahoo.com',
                                        'http://www.altavista.com',
                                        'http://www.maximumcash.com',
                                        'http://www.hitbox.com',
                                        'http://www.google.com'
                                ]
                self.outputDir = outputDir

                self.inputUrl = ""

        def isParsable(self, filename):
                if re.match("^.*[Jj][Pp][Gg]+$", filename) != None:
                        return 0
                elif re.match("^.*[Pp][Dd][Ff]+$", filename) != None:
                        return 0
                elif re.match("^.*[Gg][Ii][Ff]+$", filename) != None:
                        return 0
                elif re.match("^.*[Jj][Pp][Ee][Gg]+$", filename) != None:
                        return 0
                elif re.match("^.*[Pp][Nn][Gg]+$", filename) != None:
                        return 0
                elif re.match("^.*[Tt][Ii][Ff]+[Ff]?$", filename) != None:
                        return 0
                elif re.match("^.*[Mm][Pp]+[Ee]?[Gg]?[1234]?$", filename) != None:
                        return 0
                elif re.match("^.*[Rr]+[Aa]?[Mm]+$", filename) != None:
```

```
                return 0
        else:
                return 1

    def add_new_url(self, url):
        value = 1
        domain_value = 1

        parser = NCUrlParser(url)

        domain = parser.get_hostname()

        new_domain = "http://%s" % domain

        for x in self.bad_urls:
                if x == new_domain:
                        return

        for x in self.new_domains:
                if x == new_domain:
                        return

        self.new_domains.append(new_domain)

        for x in self.new_urls:
                if x == url:
                        return

        self.new_urls.append(url)

    def single_page_crawl(self, url):

        try:
                self.connect(url)
                val = self.parse_page()
                if val != None:
                        self.dump()
        except NCError, msg:
                raise msg



    def full_page_crawl(self, url):

        url_crawlable = 1
        new_url_crawlable = 1

        for x in self.url_cache:
                if x == url:
                        return

        if url_crawlable == 1:
            if self.isParsable(string.strip(url)):
                try:
                    print "Processing: %s" % url
                    self.single_page_crawl(url)
                    self.url_cache.append(url)
                except NCError, msg:
                    raise msg

        if len(self.web_doc.relative_urls_no_image) > 0:

            for x in self.web_doc.relative_urls_no_image:
                if not self.isParsable(string.strip(x.filename)):
                    continue
                matchStr = "^[Hh][Tt][Tt][Pp]://%s.*" + self.url_base
                if re.match(matchStr, string.strip(x.filename)):
                    newUrl = string.strip(x.filename)
                elif re.match("^[Jj][Aa][Vv][Aa][Ss][Cc][Rr][Ii][Pp][Tt]:+.*",
                string.strip(x.filename)):
                    continue
                elif re.match("^[Hh][Tt][Tt][Pp]://[^%s].*", string.strip(x.
                filename)):
                    self.add_new_url(string.strip(x.filename))
                    continue
                elif re.match("^/+.*", string.strip(x.filename)):
                    newUrl = "http://" + self.url_base + string.strip(x.
                    filename)
                elif re.match("^mailto", string.strip(x.filename)):
                    continue
                elif re.match("^[Ff][Tt][Pp]+.*", string.strip(x.filename)):
                    self.ftp_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Nn][Ee][Ww][Ss]+.*", string.strip(x.filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Gg][Oo][Pp][Hh][Ee][Rr]+.*", string.strip(x.
                filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Tt][Ee][Ll][Nn][Ee][Tt]+.*", string.strip(x.
                filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                else:
                    newUrl = "http://" + self.url_base + "/" + string.strip(
                    x.filename)

                for x in self.url_cache:
                    if x == newUrl:
                        url_crawlable = 0
                        break

                if url_crawlable == 1:
                    try:
```

```
                                        self.relative_urls.append(newUrl)
                                except NCError, msg:
                                        print msg


                        url_crawlable = 1

                if len(self.web_doc.relative_urls) > 0:

                        for x in self.web_doc.relative_urls:
                                matchStr = "^[Hh][Tt][Tt][Pp]://%s.*" + self.url_base
                                if re.match(matchStr, string.strip(x.filename)):
                                        newUrl = string.strip(x.filename)
                                elif re.match("^[Jj][Aa][Vv][Aa][Ss][Cc][Rr][Ii][Pp][Tt]:+.*",
                                string.strip(x.filename)):
                                        continue
                                elif re.match("^[Hh][Tt][Tt][Pp]://[^%s].*", string.strip(x.
                                filename)):
                                        self.add_new_url(string.strip(x.filename))
                                        continue
                                elif re.match("^/+.*", string.strip(x.filename)):
                                        newUrl = "http://" + self.url_base + string.strip(x.
                                        filename)
                                elif re.match("^mailto", string.strip(x.filename)):
                                        continue
                                elif re.match("^[Ff][Tt][Pp]+.*", string.strip(x.filename)):
                                        self.ftp_urls.append(string.strip(x.filename))
                                        continue
                                elif re.match("^[Nn][Ee][Ww][Ss]+.*", string.strip(x.filename)):
                                        self.weird_urls.append(string.strip(x.filename))
                                        continue
                                elif re.match("^[Gg][Oo][Pp][Hh][Ee][Rr]+.*", string.strip(x.
                                filename)):
                                        self.weird_urls.append(string.strip(x.filename))
                                        continue
                                elif re.match("^[Tt][Ee][Ll][Nn][Ee][Tt]+.*", string.strip(x.
                                filename)):
                                        self.weird_urls.append(string.strip(x.filename))
                                        continue
                                else:
                                        newUrl = "http://" + self.url_base + "/" + string.strip(
                                        x.filename)

                                for x in self.url_cache:
                                        if x == newUrl:
                                                url_crawlable = 0
                                                break

                                if url_crawlable == 1:
                                        try:
                                                self.relative_urls.append(newUrl)
                                        except NCError, msg:
                                                print msg
                url_crawlable = 1

        if len(self.web_doc.absolute_urls_no_image) > 0:

            for x in self.web_doc.absolute_urls_no_image:
                matchStr = "^[Hh][Tt][Tt][Pp]://%s.*" + self.url_base
                if re.match(matchStr, string.strip(x.filename)):
                    newUrl = string.strip(x.filename)
                elif re.match("^[Jj][Aa][Vv][Aa][Ss][Cc][Rr][Ii][Pp][Tt]:+.*",
                string.strip(x.filename)):
                    continue
                elif re.match("^[Hh][Tt][Tt][Pp]://[^%s].*", string.strip(x.
                filename)):
                    self.add_new_url(string.strip(x.filename))
                    continue
                #elif re.match("^/+.*", string.strip(x.filename)):
                #       newUrl = "http://" + self.url_base +
                string.strip(x.filename)
                elif re.match("^mailto", string.strip(x.filename)):
                    continue
                elif re.match("^[Ff][Tt][Pp]+.*", string.strip(x.filename)):
                    self.ftp_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Nn][Ee][Ww][Ss]+.*", string.strip(x.filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Gg][Oo][Pp][Hh][Ee][Rr]+.*", string.strip(x.
                filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Tt][Ee][Ll][Nn][Ee][Tt]+.*", string.strip(x.
                filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                #else:
                #       newUrl = "http://" + self.url_base + "/" +
                string.strip(x.filename)

                for x in self.url_cache:
                    if x == newUrl:
                        url_crawlable = 0
                        break

                if url_crawlable == 1:
                    try:
                        self.relative_urls.append(newUrl)
                    except NCError, msg:
                        print msg


                url_crawlable = 1
        if len(self.web_doc.absolute_urls) > 0:

            for x in self.web_doc.absolute_urls:
                matchStr = "^[Hh][Tt][Tt][Pp]://%s.*" + self.url_base
                if re.match(matchStr, string.strip(x.filename)):
                    newUrl = string.strip(x.filename)
                elif re.match("^[Jj][Aa][Vv][Aa][Ss][Cc][Rr][Ii][Pp][Tt]:+.*",
                string.strip(x.filename)):
                    continue
                elif re.match("^[Hh][Tt][Tt][Pp]://[^%s].*", string.strip(x.
                filename)):
                    self.add_new_url(string.strip(x.filename))
                    continue
                #elif re.match("^/+.*", string.strip(x.filename)):
                #       newUrl = "http://" + self.url_base +
                string.strip(x.filename)
                elif re.match("^mailto", string.strip(x.filename)):
                    continue
                elif re.match("^[Ff][Tt][Pp]+.*", string.strip(x.filename)):
                    self.ftp_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Nn][Ee][Ww][Ss]+.*", string.strip(x.filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Gg][Oo][Pp][Hh][Ee][Rr]+.*", string.strip(x.
                filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                elif re.match("^[Tt][Ee][Ll][Nn][Ee][Tt]+.*", string.strip(x.
                filename)):
                    self.weird_urls.append(string.strip(x.filename))
                    continue
                #else:
                #       newUrl = "http://" + self.url_base + "/" +
                string.strip(x.filename)

                for x in self.url_cache:
                    if x == newUrl:
                        url_crawlable = 0
                        break

                if url_crawlable == 1:
                    try:
                        self.relative_urls.append(newUrl)
                    except NCError, msg:
                        print msg


                url_crawlable = 1

        return
```

```
    def full_site_crawl(self, url):
        self.full_page_crawl(url)
        counter = 0

        while counter < len(self.relative_urls):
            self.full_page_crawl(self.relative_urls[counter])
            self.url_cache.append(self.relative_urls[counter])
            counter = counter + 1



    def connect(self, url):
        self.headers = None
        self.redirectCount = 0;

        self.inputUrl = url


        try:
            self.web_conn = NCHTTPInet(self.inputUrl)
            self.web_conn.connect()
        except NCError, msg:
            message = "NetCrawler Error: %s" % msg
            raise NCError(message)

        self.headers = self.web_conn.get_response_headers()
        self.url_base = self.web_conn.url_parser.get_hostname()
        self.url_path = self.web_conn.url_parser.get_path()
        self.url_port = self.web_conn.url_parser.get_port()
        self.url_query = self.web_conn.url_parser.get_query()
        self.url_query_param = self.web_conn.url_parser.get_query_param()

        while self.web_conn.get_error_code() > 299 and self.web_conn.get_error_code() <
        400 and self.redirectCount < 5:
            print "Redirecting..."
            location = self.headers['location']
            print "Redirecting to: %s" % location
            self.web_conn = NCHTTPInet(location)
            self.web_conn.connect()
            self.redirectCount = self.redirectCount + 1

        if self.redirectCount >= 5:
            message = "300 Series: %s --> %s" % (self.web_conn.get_error_code(),
            self.web_conn.get_error_message())
            raise NCError(message)

        if self.web_conn.get_error_code() > 499 and self.web_conn.get_error_code() < 600:
            message =  "500 Series: %s --> %s" % (self.web_conn.get_error_code(),
            self.web_conn.get_error_message())
            raise NCError(message)

        if self.web_conn.get_error_code() > 399 and self.web_conn.get_error_code() < 500:
            message = "400 Series: %s --> %s" % (self.web_conn.get_error_code(),
                self.web_conn.get_error_message())
                raise NCError(message)

        if self.web_conn.get_error_code() > 399 and self.web_conn.get_error_code() < 500:
                message = "200 Series: %s --> %s" % (self.web_conn.get_error_code(),
                self.web_conn.get_error_message())
                raise NCError(message)

        if self.headers != None and self.headers.has_key("content-type"):
                self.content_type = self.headers['content-type']

        if self.headers != None and self.headers.has_key("server"):
                self.server_name = self.headers['server']

        if self.headers != None and self.headers.has_key("date"):
                self.response_date = self.headers['date']

    def parse_page(self):
        # If we have gotten this far the error code was a 200 (SUCCESS!!)

        self.web_doc = NCWebDocument(self.web_conn.get_input_stream())
        if self.web_doc != None:
                val = self.web_doc.load()
                if val == None:
                        return None
                return val

    def dump(self):
        webbasedir = self.outputDir
        fileDir = webbasedir + os.sep + self.url_base

        if not os.path.exists(fileDir):
                os.mkdir(fileDir)

        filename = string.replace(self.url_path, "/",".")


        if filename == ".":
                full_filename =  fileDir + os.sep + ".root" + ".xml"
        else:
                full_filename = fileDir + os.sep + filename + ".xml"

        if self.outputDir == "":
                outputFile = sys.stdout
        else:
                outputFile = open(full_filename, "w+")

        outputFile.write("<?xml version='1.0' ?>\n")
        outputFile.write( "<WebDocument>\n")
        outputData = "<Title>%s</Title>\n" % self.web_doc.title_text
        outputFile.write(outputData)

        for x in self.headers.keys():
```

```
                                outputData =  "<%s>%s</%s>\n" % (x, self.headers[x], x)
                                outputFile.write(outputData)

                        if self.url_base != '':
                                outputData =  "<UrlBase>%s</UrlBase>\n" % self.url_base
                                outputFile.write(outputData)

                        if self.url_path != '':
                                outputData =  "<UrlPath>%s</UrlPath>\n" % self.url_path
                                outputFile.write(outputData)

                        if self.url_port != '':
                                outputData = "<UrlPort>%s</UrlPort>\n" % self.url_port
                                outputFile.write(outputData)

                        if self.url_query != '':
                                outputData = "<UrlQuery>%s</UrlQuery>\n" % self.url_query
                                outputFile.write(outputData)

                        if self.url_query_param != '':
                                outputData = "<UrlQueryParam>%s</UrlQueryParam>\n" % self.url_query_param
                                outputFile.write(outputData)

                        if len(self.web_doc.meta_tags) > 0:
                                outputData = "<MetaTags count='%d'>\n" % len(self.web_doc.meta_tags)
                                outputFile.write(outputData)
                                for x in self.web_doc.meta_tags:
                                        if x != None:
                                                outputFile.write(x.get_xml())
                                outputFile.write("</MetaTags>\n")

                        if len(self.web_doc.frameset_tags) > 0:
                                outputData = "<Framesets count='%d'>\n" % len(self.web_doc.frameset_tags)
                                outputFile.write(outputData)
                                for x in self.web_doc.frameset_tags:
                                        if x != None:
                                                outputFile.write(x.get_xml())
                                outputFile.write("</Framesets>\n")

                        if len(self.web_doc.absolute_urls_no_image) > 0:
                                outputData = "<AbsoluteUrlsNoImage count='%d'>\n" % len(self.web_doc.
                                absolute_urls_no_image)
                                outputFile.write(outputData)
                                for x in self.web_doc.absolute_urls_no_image:
                                        if x != None:
                                                outputFile.write(x.get_xml())
                                outputFile.write( "</AbsoluteUrlsNoImage>\n")

                        if len(self.web_doc.absolute_urls) > 0:
                                outputData = "<AbsoluteUrls count='%d'>\m" % len(self.web_doc.
                                absolute_urls)
                                outputFile.write(outputData)
                                for x in self.web_doc.absolute_urls:
```

```
                if x != None:
                        outputFile.write(x.get_xml())
        outputFile.write( "</AbsoluteUrls>\n")

if len(self.web_doc.relative_urls_no_image) > 0:
        outputData = "<RelativeUrlsNoImage count='%d'>\n"  % len(self.web_doc.
        relative_urls_no_image)
        outputFile.write(outputData)
        for x in self.web_doc.relative_urls_no_image:
                if x != None:
                        outputFile.write( x.get_xml())
        outputFile.write( "</RelativeUrlsNoImage>\n")

if len(self.web_doc.relative_urls) > 0:
        outputData = "<RelativeUrls count='%d'>\m" % len(self.web_doc.
        relative_urls)
        outputFile.write(outputData)
        for x in self.web_doc.relative_urls:
                if x != None:
                        outputFile.write( x.get_xml())
        outputFile.write( "</RelativeUrls>\n")

if len(self.web_doc.centered_text) > 0:
        outputData = "<CenterText count='%d'>\n" % len(self.web_doc.
        centered_text)
        outputFile.write(outputData)
        for x in self.web_doc.centered_text:
                outputData = "\t<CenterTextItem>\n\t\t%s\n\t</CenterTextItem>\n"
                 % x
        outputFile.write(outputData)
        outputFile.write( "</CenterText>\n")

if len(self.web_doc.span_text) > 0:
        outputData = "<SpanText count='%d'>\n" % len(self.web_doc.span_text)
        outputFile.write(outputData)
        for x in self.web_doc.span_text:
                outputData = "\t<SpanTextItem>\n\t\t%s\n\t</SpanTextItem>\n" % x
                outputFile.write(outputData)
        outputFile.write( "</SpanText>\n")

if len(self.web_doc.table_text) > 0:
        outputData = "<TableText count='%d'>" % len(self.web_doc.table_text)
        outputFile.write(outputData)
        for x in self.web_doc.table_text:
                outputData = "\t<TableTextItem>\n\t\t%s\n\t</TableTextItem>\n" %
                 x
                outputFile.write(outputData)
        outputFile.write( "</TableText>")

if len(self.web_doc.caption_text) > 0:
        outputData = "<Captions count='%d'>\n" % len(self.web_doc.caption_text)
        outputFile.write(outputData)
        for x in self.web_doc.caption_text:
```

```
                    outputData = "\t<CaptionItem>\n\t\t%s\n\t</CaptionItem>\n" % x
                    outputFile.write(outputData)
                outputFile.write( "</Captions>\n")

        if len(self.web_doc.paragraph_text) > 0:
                outputData = "<ParagraphText count='%d'>\n" % len(self.web_doc.
                paragraph_text)
                outputFile.write(outputData)
                for x in self.web_doc.paragraph_text:
                    outputData = "\t<ParagraphItem>\n\t\t%s\n\t</ParagraphItem>\n" %
                     x
                    outputFile.write(outputData)
                outputFile.write("</ParagraphText>\n")

        if len(self.web_doc.tr_text) > 0:
                outputData = "<TableRowText count='%d'>" % len(self.web_doc.tr_text)
                outputFile.write(outputData)
                for x in self.web_doc.tr_text:
                    outputData = "\t<TableRowItem>\n\t\t%s\n\t</TableRowItem>\n" % x
                    outputFile.write(outputData)
                outputFile.write( "</TableRowText>")

        if len(self.web_doc.td_text) > 0:
                outputData = "<TableDataText count='%d'>\n" % len(self.web_doc.td_text)
                outputFile.write(outputData)
                for x in self.web_doc.td_text:
                    outputData = "\t<TableDataItem>\n\t\t%s\n\t</TableDataItem>\n" %
                     x
                    outputFile.write(outputData)
                outputFile.write( "</TableDataText>\n")

        if len(self.web_doc.th_text) > 0:
                outputData = "<TableHeaderText count='%d'>\n" % len(self.web_doc.th_text)
                outputFile.write(outputData)
                for x in self.web_doc.th_text:
                    outputData =
                    "\t<TableHeaderItem>\n\t\t%s\n\t</TableHeaderItem>\n" % x
                    outputFile.write(outputData)
                outputFile.write( "</TableHeaderText>\n")

        if len(self.web_doc.script_tags) > 0:
                outputData = "<Script count='%d'>\n" % len(self.web_doc.script_tags)
                outputFile.write(outputData)
                for x in self.web_doc.script_tags:
                    outputData = "\t<ScriptItem>\n\t\t%s\n\t</ScriptItem>\n" % x
                    outputFile.write(outputData)
                outputFile.write( "</Script>\n")

        outputFile.write( "</WebDocument>\n")

    def reset(self):
        self.headers = None
        self.redirectCount = 0;
```

```
                self.url_cache = []
                self.outputDir = outputDir

                self.inputUrl = ""
                self.web_doc = None



if __name__ == '__main__':

        headers = None

        if len(sys.argv) < 2:
                print "Usage: NetCrawler.py <url> <output directory>"
                print
                sys.exit(1)

        inputUrl = sys.argv[1]
        outputDir = sys.argv[2]
        url_crawlable = 1

        newUrlCache = []
        url_cache = []

        crawler = NetCrawler(outputDir)

        try:
                crawler.full_site_crawl(inputUrl)
        except NCError, msg:
                print msg

        except KeyboardInterrupt:
                print "Exiting..."
                sys.exit(1)

        for x in crawler.web_doc.script_tags:
                crawler.add_new_url(x)

        for x in crawler.new_domains:
                print "New Domains: %s" % x

        for x in crawler.new_urls:
                print "New Urls: %s" % x


        sys.exit(0)
```