CIT 483/583 Lab –OpenURI, & Nokogiri

**Instructions**

Open and save this file in any MS word-compatible format as Lab07\_*Firstname*\_*Lastname*.<ext> and place your answers in that document. Do a **Save-As** and retain all of my content. Keep the document safe in case your submission fails, or you discover an error prior to the due date and wish to re-submit. Submit your document to the Lab07 dropbox in Canvas. The due date and any other pertinent information are noted in the Canvas item.

This lab should be completed on students.cs.nku.edu as a reference implementation and as a means of testing your answers.

**Place the answers in or immediately following each question and make sure your answers stand out from the questions by using a different font color.**

## OpenURI & Nokogiri

1. OpenUri can open any URI, which it treats as an IO object. It is interesting to see what happens if the resource is not a plain text file. Perform the following Linux command line interactions and capture any output, **but please abbreviate long blocks of text to the last few lines and don't worry if a command produces no output**.

$ url="https://www.nku.edu/content/dam/nkuhome/images/homefeatures/griffin-hall-fb.jpg"

$ ruby -ropen-uri -e "puts open('$url')" > picture.jpg

$ wget $url

--2018-11-01 15:52:22-- https://www.nku.edu/content/dam/nkuhome/images/homefeatures/griffin-hall-fb.jpg

Resolving www.nku.edu (www.nku.edu)... 10.2.109.51

Connecting to www.nku.edu (www.nku.edu)|10.2.109.51|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 174151 (170K) [image/jpeg]

Saving to: ‘griffin-hall-fb.jpg’

griffin-hall-fb.jpg 100%[===================>] 170.07K --.-KB/s in 0.003s

2018-11-01 15:52:23 (65.6 MB/s) - ‘griffin-hall-fb.jpg’ saved [174151/174151]

2015-11-21 13:02:38 (54.3 MB/s) - ‘banner.jpg.1’ saved [28662/28662]

$ file \*jpg

griffin-hall-fb.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=12, height=900, bps=0, PhotometricIntepretation=RGB, orientation=upper-left, width=1200], baseline, precision 8, 1200x628, frames 3

picture.jpg: ASCII tex

(a) Inspect the output of the Ruby one-liner directly (don't redirect to a file).

$ ruby -ropen-uri -e "puts open('$url')"

#<File:0x000000029f45f0>

This explains why picture.jpg is not really a graphics file. It is a File object, and we need to process it as a file to see its contents.

(b) We need to tweak the command to generate the *contents* of the file.

$ ruby -ropen-uri -e "puts open('$url').read" | more

[Binary output] Ctrl-C to end

Hit return a few times and then Ctrl-C. Clearly, this is not line-oriented text.

(c) This seems to be binary data, so let’s recreate the first attempt to generate a .jpg file.

$ ruby -ropen-uri -e "puts open('$url').read" > picture.jpg

$ cmp -b -l picture.jpg banner.jpg

cmp: EOF on griffin-hall-fb.jpg

(d) Since puts adds a newline, let's see if that is the difference.

$ ruby -ropen-uri -e "print open('$url').read" > picture.jpg

$ ls -l \*jpg

-rw-r--r-- 1 mccordt domain^users 174151 Nov 1 15:51 griffin-hall-fb.jpg

-rw-r--r-- 1 mccordt domain^users 174152 Nov 1 15:56 picture.jpg

$ cmp -b -l picture.jpg banner.jpg # no output means they are the same

[No output]

1. Perform the following interactions in irb. Capture the output of each command. **Abbreviate any output that exceeds 8-10 lines**.

$ irb --simple-prompt

>> require 'open-uri'

=> true

require 'nokogiri'

=> true

Contents as of 11/01/2018 – Could change over time

>> doc = open ("https://inside.nku.edu/informatics/departments/computerscience.html")

=> #<Tempfile:/tmp/open-uri20181101-17771-1nlcbtt>

>> page = Nokogiri::HTML(doc)

…

::Element:0xb30254 name="br">, #<Nokogiri::XML::Text:0xb30060 "\n">, #<Nokogiri::XML::Element:0xb2ff84 name="div" children=[#<Nokogiri::XML::Text:0xb2fda4 "© 2018 Northern Kentucky University. All rights reserved.">]>, #<Nokogiri::XML::Text:0xb2fc00 "\n">]>, #<Nokogiri::XML::Text:0xb2fa5c "\n">]>, #<Nokogiri::XML::Text:0xb2f87c "\n">]>, #<Nokogiri::XML::Text:0xb2f6d8 "\n">]>, #<Nokogiri::XML::Text:0xb2f50c "\n">]>, #<Nokogiri::XML::Text:0xb2f340 "\n">]>, #<Nokogiri::XML::Text:0xb2f19c "\r\n">]>, #<Nokogiri::XML::Text:0xb2ef6c "\r\n">]>, #<Nokogiri::XML::Text:0xb2edc8 "\n">]>, #<Nokogiri::XML::Text:0xb2ec24 "\n">]>, #<Nokogiri::XML::Text:0xb2ea80 "\n">]>, #<Nokogiri::XML::Text:0xb2e8dc "\n">]>, #<Nokogiri::XML::Text:0xb2e738 "\n">]>, #<Nokogiri::XML::Text:0xb2e544 "\n\n">]>, #<Nokogiri::XML::Text:0xb2e3a0 "\n">]>]>

>> page.css('img').each { |img| puts img['src'] }

/content/dam/administrative-pages/images/canvas.png

/content/dam/administrative-pages/images/calendar.png

/content/dam/administrative-pages/images/nku-catalog.png

/content/dam/administrative-pages/images/directory.png

/content/dam/administrative-pages/images/library.png

/content/dam/administrative-pages/images/myNKU.png

/content/dam/administrative-pages/images/sync.png

/content/dam/administrative-pages/images/shop.png

/content/dam/administrative-pages/images/mail.png

/content/dam/administrative-pages/images/nku-change-password.png

/content/dam/administrative-pages/images/nku-azlist.png

/content/dam/www/images/logo.png

. . .

>> page.css('div').collect { |div| div['class'] }.uniq

=> ["header iparsys parsys", "section", "new", "iparys\_inherited", "reference pa rbase section", "cq-dd-paragraph", "text parbase", "text ", nil, "container", "r ow", "col-xs-4 col-md-2", "navbar navbar-default nku-main-navbar", "container-fl uid", "navbar-header", "dropdown", "visible-xs", "input-group", "input-group-btn ", "navbar-collapse collapse", "site\_search search\_form", "columncontrol section ", "row nku-gold-background nku-inside-banner", "col-md-12 ", "container-fluid n ku-main", "banner parsys", "textimage parbase section", "textimage nku-interior- header nku-headline", "image topimage nku-interior-header nku-headlineImage", "t ext nku-interior-header nku-headline Text", "clear", "col-md-12", "breadcrumb", "breadcrumb-disabled", "pagetitle title", "par parsys", "text parbase section", "row ", "col-sm-3 ", "nkusidenav section", "none", "nku-side-nav-mobile-toggle", "col-sm-8 ", "slideshow parbase section", "nku-photo-gallery-list hidden nku-pe ople-directory nku-photo-gallery-list-100-140 ", "nku-photo-gallery-list-item ", "image", "text", "nku-photo-gallery-list-description", "nku-modal", "col-sm-1 " , "footer iparsys parsys", "col-md-6 border-left col-xs-6", "col-md-6 col-xs-6", "col-md-7 col-md-push-5", "col-md-6 col-md-push-6", "row footer-nav-links", "co l-md-6", "col-md-6 col-md-pull-6 border-left-right", "col-md-5 col-md-pull-7", " col-md-4 hidden-sm hidden-xs", "col-md-8 col-xs-12"]

>> page.css('a').collect { |a| a['href'] }.each { |href| puts href }; p

https://inside.nku.edu/cite/canvasblackboard.html

https://www.nku.edu/calendars.html

https://inside.nku.edu/registrar/catalog.html

https://directory.nku.edu

https://inside.nku.edu/library.html

https://mynku.nku.edu

https://nku.campuslabs.com/engage/

https://inside.nku.edu/shop.html

https://inside.nku.edu/it/webmail.html

https://password.nku.edu

https://www.nku.edu/azlisting.html

https://www.nku.edu/

#quicklinks

<https://www.nku.edu/about.html>

. . .

(a) How many total anchor tags ("a") are in the document?

page.css('a').to\_a.size

=> 154

(b) How many unique 'href's are used within those anchor tags?

page.css('a').collect { |a| a['href'] }.uniq.size

=> 142

(c) How many image ('img') tags are in the document?

page.css('img').to\_a.size

=> 48

(d) How many unique image sources ('src') are used?

>> page.css('img').collect { |i| i['src'] }.uniq.size

=> 47

1. Nokogiri can also process XML files. If you use the same IRB session, you will have the required libraries, but if not, make sure you add those libraries to a new session. **Make sure that the file master\_scoreboard.xml from an earlier lab is in the directory in which IRB is running.** As usual, capture the output of all commands, but abbreviate long blocks to the last few lines.

$ irb --simple-prompt

>> require 'open-uri'

require 'nokogiri'

>> raw = open("master\_scoreboard.xml")

=> #<File:master\_scoreboard.xml>

>> raw.readpartial(80) # read 80 chars

=> "<?xml version=\"1.0\" encoding=\"UTF-8\"?><!--Copyright 2016 MLB Advanced Media, L.P"

>> doc = Nokogiri::XML(open('master\_scoreboard.xml'))

. . .

ta\_directory" value="/components/game/mlb/year\_2015/month\_07/day\_19/gid\_2015\_07\_19\_bosmlb\_anamlb\_1">, #<Nokogiri::XML::Attr:0xcd0118 name="league" value="AA">] children=[#<Nokogiri::XML::Text:0xc90414 "\n ">, #<Nokogiri::XML::Element:0xc9034c name="status" attributes=[#<Nokogiri::XML::Attr:0xc902d4 name="status" value="Postponed">, #<Nokogiri::XML::Attr:0xc902c0 name="ind" value="DR">, #<Nokogiri::XML::Attr:0xc902ac name="reason" value="Rain">, #<Nokogiri::XML::Attr:0xc90298 name="inning\_state">, #<Nokogiri::XML::Attr:0xc90270 name="note">]>, #<Nokogiri::XML::Text:0xc8d64c "\n ">, #<Nokogiri::XML::Element:0xc8d598 name="game\_media">, #<Nokogiri::XML::Text:0xc8d3cc "\n ">]>, #<Nokogiri::XML::Text:0xc8d214 "\n">]>]>

>> doc.xpath('//game').each do |p|

puts p.attributes['id']

end

2015/07/19/seamlb-nyamlb-1

2015/07/19/tbamlb-tormlb-1

2015/07/19/balmlb-detmlb-1

2015/07/19/clemlb-cinmlb-1

2015/07/19/lanmlb-wasmlb-1

2015/07/19/miamlb-phimlb-1

2015/07/19/kcamlb-chamlb-1

2015/07/19/pitmlb-milmlb-1

2015/07/19/texmlb-houmlb-1

2015/07/19/nynmlb-slnmlb-1

2015/07/19/minmlb-oakmlb-1

2015/07/19/colmlb-sdnmlb-1

2015/07/19/sfnmlb-arimlb-1

2015/07/19/chnmlb-atlmlb-1

2015/07/19/bosmlb-anamlb-1

=> 0

>> doc.xpath('//winning\_pitcher').each do |p|

puts p.attributes['last']

end

Betances

Estrada

Gonzalez

Rzepczynski

Greinke

Neris

Duffy

Jungmann

Keuchel

Torres

Chavez

Bumgarner

Arrieta

As mentioned in the slides, the slop method can generate objects/methods for elements, but in a limited way. Nested elements are treated as if they are children of higher-level elements.

>> puts doc.games # should error out

NoMethodError: undefined method `games' for #<Nokogiri::XML::Document:0x0000000000ed8a50>

from (irb):26

from /usr/bin/irb:11:in `<main>'

>> doc = Nokogiri::XML(open('master\_scoreboard.xml')).slop!; p

=> nil

>> puts doc.games.class # games now returns an element

Nokogiri::XML::Element

=> nil

>> puts doc.games.game

. . .

<game id="2015/07/19/bosmlb-anamlb-1" venue="Angel Stadium of Anaheim" game\_pk="415020" time="8:05" time\_date="2015/07/19 8:05" time\_date\_aw\_lg="2015/07/19 8:05" time\_date\_hm\_lg="2015/07/19 8:05" time\_zone="ET" ampm="PM" first\_pitch\_et="" away\_time="8:05" away\_time\_zone="ET" away\_ampm="PM" home\_time="5:05" home\_time\_zone="PT" home\_ampm="PM" game\_type="R" tiebreaker\_sw="N" resume\_date="" original\_date="2015/07/19" time\_zone\_aw\_lg="-4" time\_zone\_hm\_lg="-4" time\_aw\_lg="8:05" aw\_lg\_ampm="PM" tz\_aw\_lg\_gen="ET" time\_hm\_lg="8:05" hm\_lg\_ampm="PM" tz\_hm\_lg\_gen="ET" venue\_id="1" scheduled\_innings="9" description="" away\_name\_abbrev="BOS" home\_name\_abbrev="LAA" away\_code="bos" away\_file\_code="bos" away\_team\_id="111" away\_team\_city="Boston" away\_team\_name="Red Sox" away\_division="E" away\_league\_id="103" away\_sport\_code="mlb" home\_code="ana" home\_file\_code="ana" home\_team\_id="108" home\_team\_city="LA Angels" home\_team\_name="Angels" home\_division="W" home\_league\_id="103" home\_sport\_code="mlb" day="SUN" gameday\_sw="P" double\_header\_sw="N" game\_nbr="1" tbd\_flag="N" away\_games\_back="8.0" home\_games\_back="-" away\_games\_back\_wildcard="7.0" home\_games\_back\_wildcard="" venue\_w\_chan\_loc="USCA0027" location="Anaheim, CA" gameday="2015\_07\_19\_bosmlb\_anamlb\_1" away\_win="42" away\_loss="49" home\_win="50" home\_loss="40" game\_data\_directory="/components/game/mlb/year\_2015/month\_07/day\_19/gid\_2015\_07\_19\_bosmlb\_anamlb\_1" league="AA">

<status status="Postponed" ind="DR" reason="Rain" inning\_state="" note=""/>

<game\_media/>

</game>

=> nil

>> doc.games.game.each do |p|

puts p.attributes['id']

end; p

2015/07/19/seamlb-nyamlb-1

2015/07/19/tbamlb-tormlb-1

2015/07/19/balmlb-detmlb-1

2015/07/19/clemlb-cinmlb-1

2015/07/19/lanmlb-wasmlb-1

2015/07/19/miamlb-phimlb-1

2015/07/19/kcamlb-chamlb-1

2015/07/19/pitmlb-milmlb-1

2015/07/19/texmlb-houmlb-1

2015/07/19/nynmlb-slnmlb-1

2015/07/19/minmlb-oakmlb-1

2015/07/19/colmlb-sdnmlb-1

2015/07/19/sfnmlb-arimlb-1

2015/07/19/chnmlb-atlmlb-1

2015/07/19/bosmlb-anamlb-1

=> nil

1. Copy ~mccordt/find\_links.rb find\_links.rb on students.cs.nku.edu to a location that you can write to. For reference, here is the content:

#!/usr/bin/ruby

# file: find\_links.rb

require 'open-uri'

require 'nokogiri'

if ARGV.length == 0

$stderr.puts "Must specify a URL"

exit 1

end

ARGV.each do |arg|

open(arg) do |f|

# Display connection data

puts "#"\*25 + "\nConnection: '#{arg}'\n" + "#"\*25

[:base\_uri, :meta, :status, :charset, :content\_encoding,

:content\_type, :last\_modi\_ed].each do |method|

puts "#{method.to\_s}: #{f.send(method)}" if f.respond\_to? method

end

# Display the anchor links

Nokogiri::HTML(f).css('a').each do |anchor|

href = anchor['href']

puts href

end

end # open(arg)

end # ARGv.each

(a) Run the script with the following argument and show the abbreviated output:

$ ruby find\_links.rb <http://informatics.nku.edu/departments/computer-science.html>

#########################

Connection: 'https://inside.nku.edu/informatics/departments/computerscience.html'

#########################

base\_uri: https://inside.nku.edu/informatics/departments/computerscience.html

meta: {"date"=>"Thu, 01 Nov 2018 20:17:21 GMT", "server"=>"Apache/2.4.6 (Red Hat Enterprise Linux) OpenSSL/1.0.2k-fips Communique/4.1.12", "vary"=>"Host", "last-modified"=>"Thu, 01 Nov 2018 20:17:21 GMT", "etag"=>"W/\"1646b-579a01b381226\"", "accept-ranges"=>"bytes", "content-length"=>"91243", "content-type"=>"text/html"}

status: ["200", "OK"]

charset:

content\_encoding: []

content\_type: text/html

https://inside.nku.edu/cite/canvasblackboard.html

https://www.nku.edu/calendars.html

https://inside.nku.edu/registrar/catalog.html

https://directory.nku.edu

https://inside.nku.edu/library.html

https://mynku.nku.edu

<https://nku.campuslabs.com/engage/>

. . .

(b) Many of the hrefs displayed by the script are relative. For the hrefs beginning with a '/', we should prepend the host's URL. Add the following line (before the call to Nokogiri) to obtain the necessary base url text.

host\_url = /^(.\*\.nku\.edu)\//.match(f.base\_uri.to\_s)[1]

Now modify the code to prepend this text to hrefs that begin with a '/' and .

Nokogiri::HTML(f).css('a').each do |anchor|

href = anchor['href']

# type your added code here

puts href

end

(c) Add the following code find those hrefs that may be problematic for users.

Nokogiri::HTML(f).css('a').each do |anchor|

href = anchor['href']

# code from (b) here

# puts href # comment out the puts

if /^https?:\/\// === href

begin

puts href unless open(href).status[0] == "200"

rescue Exception => e

puts "ERROR: #{e} --#{href}"

end

end

Complete code:

#!/usr/bin/ruby

require 'open-uri'

require 'nokogiri'

if ARGV.length == 0

$stderr.puts "Must specify a URL"

exit 1

end

ARGV.each do |arg|

open(arg) do |f|

# Display connection data

puts "#"\*25 + "\nConnection: '#{arg}'\n" + "#"\*25

[:base\_uri, :meta, :status, :charset, :content\_encoding,

:content\_type, :last\_modified].each do |method|

puts "#{method.to\_s}: #{f.send(method)}" if f.respond\_to? method

end

# Display the anchor links

base\_url = /^(.\*\.nku\.edu)\//.match(f.base\_uri.to\_s)[1]

puts "host\_url: #{host\_url}"

Nokogiri::HTML(f).css('a').each do |anchor|

href = anchor['href']

# puts href

if /^https?:\/\// === href

begin

puts href unless open(href).status[0] == "200"

rescue Exception => e

puts "ERROR: #{e} --#{href}"

end

end

end

end # open(arg)

end # ARGV.each

List all of the problematic links you found.

ERROR: SSL\_connect SYSCALL returned=5 errno=0 state=unknown state --https://supportnku.nku.edu/SSLPage.aspx?pid=234

ERROR: SSL\_connect SYSCALL returned=5 errno=0 state=unknown state --https://supportnku.nku.edu/SSLPage.aspx?pid=234

ERROR: redirection forbidden: http://www.youtube.com/user/InformaticsNKU -> https://www.youtube.com/user/InformaticsNKU --http://www.youtube.com/user/InformaticsNKU

6. Perform the following irb interactions to locate the source code for OpenURI's open method. Capture the output of each command.

$ irb --simple-prompt

>> method(:open).owner

=> Kernel

>> method(:open).source\_location

=> nil

>> require 'open-uri'

=> true

>> method(:open).owner

=> Kernel

>> method(:open).source\_location

=> ["/usr/lib/ruby/2.1.0/open-uri.rb", 28]

## Sockets

For this part, you should open two terminals, both connecting to students.cs.nku.edu (or you may try byobu with two terminal sessions in one puTTY session.)

Create the following Ruby program. (You may copy it from students.cs.nku.edu @ ~mccordt/simple-server.rb or cut and paste into vi.)

#!/usr/bin/ruby

# simple-server.rb

# demo for a Ruby echo server

require 'socket'

# set up optional args

ip\_re = /^(?:\d{1,3}\.){3}\d{1,3}$/

ip, port = ['0.0.0.0', 1024 + rand(20000)]

ip = ARGV[0] if ip\_re === ARGV[0]

port = ARGV[1].to\_i if /\d+/ === ARGV[1]

[:ip, :port].each do |var|

puts "Variable: #{var}=#{eval var.to\_s}"

end

server = TCPServer.new(ip, port)

# loop infinitely, processing one incoming message per iteration

socket = server.accept

loop do

request = socket.gets

STDERR.puts "Received: #{request}"

socket.print "Request was: #{request}"

break if ( !request || request.chomp == 'bye' )

end

# cleanup

socket.close

1. For each of the following record the output of the command. If it fails, record the exception and backtrace. (If it succeeds, use Ctrl-C to kill the process, as it is in an infinite loop listening for clients.) Since there are multiple students doing the lab, add the day of the month of your birthday to the number 20072 used in the samples, or use another random number in the range 0-30.

$ ruby simple-server.rb 127.0.0.1 22

Variable: ip=127.0.0.1

Variable: port=22

simple-server.rb:18:in `initialize': Permission denied - bind(2) for "127.0.0.1" port 22 (Errno::EACCES)

from simple-server.rb:18:in `new'

from simple-server.rb:18:in `<main>'

$ ruby simple-server.rb 127.0.0.1 20072

Variable: ip=127.0.0.1

Variable: port=20072

$ ruby simple-server.rb students.cs.nku.edu 20072

Variable: ip=0.0.0.0

Variable: port=20072

$ ruby simple-server.rb localhost 20072

Variable: ip=0.0.0.0

Variable: port=20072

$ ruby simple-server.rb 327.5.5.5 20072

Variable: ip=327.5.5.5

Variable: port=20072

simple-server.rb:18:in `initialize': getaddrinfo: Name or service not known (SocketError)

from simple-server.rb:18:in `new'

from simple-server.rb:18:in `<main>'

2. For the next section run the commands in Courier font and answer questions a thru e.

$ ruby simple-server.rb 127.0.0.1 78004

Variable: ip=127.0.0.1

Variable: port=78004

[If the command throws an exception similar to `initialize': Address already in use increment the number by a small amount and try again.]

1. What appears to be wrong with this set of connection arguments (assuming the port is not in use)?

The port is greater than the largest number than can be stored in a 16-bit signed value.

1. Despite that, did it succeed?

Yes

1. In a second terminal while the first command is still running, enter the following command. Try to determine which port the server is actually listening on and enter it as the answer:

$ lsof -iTCP -sTCP:LISTEN | grep $USER

12468

1. Enter the following command (changing the original number 78004 as needed if you found that the original port was blocked) and record the result:

$ ruby -e 'p 78004 % 2\*\*16'

12468

1. What does Ruby do when a requested port number exceeds 216 -1?

It uses the % operator (port\_number % 65536) to bring it into the appropriate range.

For the next section, run the following paired commands on two terminals to observe whether the processes succeed in communicating. [When the port number is specified, modify the port numbers as needed in the event that port happens to be blocked by another student's process.]

If you don't specify a port, the script chooses one at random. You can use the port displayed on terminal "A" to determine the port to specify on terminal "B". You *may* need to use Ctrl-C to kill one or more processes on terminals "A" or "B". Put "Success" or "Failure" as the answer to each item a thru c and follow the instructions for letter d. Again, if the port is in use, add a small random number to it to avoid port conflicts.

a) Success

(TERM A) $ ruby simple-server.rb 127.0.0.1 9294

(TERM B) $ cat /etc/passwd | netcat 127.0.0.1 9294

b) Success

(TERM A) $ ruby simple-server.rb 127.5.0.1 3241

TERM B) $ cat /etc/passwd | netcat 127.0.5.1 3241

c) Success

TERM A) $ ruby simple-server.rb localhost 5404

TERM B) $ echo "Hello" | netcat localhost 5404

d) Capture the actual output of TERM B for this question. Type the individual lines into TERM B separated by the Enter key.

TERM A) $ ruby simple-server.rb # use the port number shown for TERM B

TERM B) netcat 172.28.100.236 <port from TERM A>

asdf

hi

Hello

Goodbye

bye

mccordt@vir:~$ netcat 172.28.100.236 15546

asdf

Request was: asdf

hi

Request was: hi

Hello

Request was: Hello

Goodbye

Request was: Goodbye

bye

Request was: bye