CSC 523 Scripting for Data Science, Fall 2023

Dr. Dale E. Parson, Assignment 1, Programming with Python regular expressions & CSV files.

DUE By 11:59 PM on Thursday September 28, 2023 via make turnitin on acad. The standard 10% per day deduction for late assignments applies. I am sending this out Labor Day weekend (no class on 9/4) with an in-class overview, Q&A, and work session planned for 9/11.

The input data file is a trace of my Ethernet activity one morning. I have extracted Ethernet link-layer data into one output comma-separated-value file. You will extract TCP/IP and UDP/IP data from the same input file into a separate CSV file. You will also extract TCP-connection-stream data into a third output file. There is a lot about network protocols embedded in the input data, but the good news is that if it is new to you, you can concentrate on the regular expressions using my Ethernet example code.

Perform the following steps to set up for this project. Start out in your login directory on csit (a.k.a. acad).

First, edit a file called **.bash_profile** in your login directory, creating it if it does not exist, and add the following lines at the bottom. Make sure to use plain ascii double quotes. MS Word may change them. **TYPE THESE LINES, DON'T COPY & PASTE THEM**.

```
# NEWLY ADDED FALL 2020 for the new acad & mcgonagall, D. Parson
export PATH="/usr/local/bin:${PATH}"
alias python="/usr/local/bin/python3.7"
alias ipython="/usr/local/bin/ipython3"
```

Save the file **.bash_profile**, log out, and log back into acad's login directory.

```
cd  $HOME
mkdir  DataMine  # This may already be there.
cd  ./DataMine
cp  ~parson/DataMine/csc523F23TCPUDP.problem.zip  csc523F23TCPUDP.problem.zip
unzip  csc523F23TCPUDP.problem.zip
cd  ./csc523F23TCPUDP
```

This is the directory from which you must run **make turnitin** by the project deadline to avoid a 10% per day late penalty. **Please do not change the name of this directory, since my test scripts depend on it.** All of the larger data files for this project are in directory ~parson/DataMine.

~parson/DataMine/WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt.gz
~parson/DataMine/Ethout.csv.ref
~parson/DataMine/TCPUDPout.csv.ref
~parson/DataMine/TCPStreams.csv.ref

All of the handout files in this project are available via the following URLs for students planning to work on home machines. You will need to move your csc523F23TCPUDP.py file back into the project directory on acad / mcgonagall for final testing and turning in. Be careful not to over-write your work. Instructions for testing at home are at the bottom of this handout.

csc523F23TCPUDP.problem.zip	LINK to handout code
csc523F23TCPUDP.py	LINK to your Python file in that zip archive
WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt.gz	LINK to handout data
WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt	LINK to plain text data
Ethout.csv.ref	LINK to output reference data file for Ethernet link layer 2 https://notes.shichao.io/tcpv1/ch3/ See example of my Ethernet matching pattern on the second-last page below.
TCPUDPout.csv.ref	LINK to output reference data file for IP layer 3 & UDP or TCP layer 4 https://notes.shichao.io/tcpv1/ch5/ https://notes.shichao.io/tcpv1/ch10/ https://notes.shichao.io/tcpv1/ch12/
TCPStreams.csv.ref	LINK to aggregated TCP client<->server communication "thread" https://notes.shichao.io/tcpv1/ch13/
testdiff.py	LINK to stay-at-home, DIY "Rapid Test"
csc523sept2023.parts6and8.py	LINK to a mini-tutorial on aggregating TCP streams in this assignment

You will see the following files in this **csc523F23TCPUDP** directory on acad / mcgonagall:

csc523F23TCPUDP.py	The Python program you must complete. We will go over August 31.
makefile	Files needed to **make test** and **make turnitin** to get your solution to me.
makelib	
textdiff.py	Python script for testing at home.

You can work on this project by copying and unzipping **csc523F23TCPUDP.problem.zip** on your own machine, or by downloading the above links. but in that case you will also need to copy these files from acad. I recommend getting local copies if possible, just so you can visually inspect them as we will do in class. You can also get them via the above LINKs.
~parson/DataMine/WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt.gz
~parson/DataMine/Ethout.csv.ref
~parson/DataMine/TCPUDPout.csv.ref
~parson/DataMine/TCPStreams.csv.ref

You are free to use ipython or Jupyter or whatever development environment you are already used to on your home machine, but I cannot support installing and configuring various miscellaneous environments because that support introduces too many variables into project specs and steps. My recommendation is to use /usr/local/bin/python3.7 and /usr/local/bin/ipython3 on acad & mcgonagall as aliased above if you are new to these servers. ALL testing this semester must pass **make clean test** in the handout acad project directory. Turn in assignments by invoking **make turnitin**.

I will demo a successful **make clean test** in class.

The handout **make clean test** looks like this:
$ **make clean test**
/bin/rm -f *.o *.class .jar core *.exe *.obj *.pyc __pycache__/*.pyc

/bin/rm -f junk* *.pyc WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt
/bin/rm -f *.out *.o *.dif *.out *.csv /home/kutztown.edu/parson/tmp/parson*csv
/bin/bash -c "PYTHONPATH=/home/kutztown.edu/parson/DataMine:.:.. /usr/local/bin/python3.7 csc523F23TCPUDP.py /home/kutztown.edu/parson/DataMine/WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt.gz /home/kutztown.edu/parson/tmp/parson_TCPUDPout.csv /home/kutztown.edu/parson/tmp/parson_TCPStreams.csv > /home/kutztown.edu/parson/tmp/parson_Ethout.csv"
/bin/ln -s /home/kutztown.edu/parson/tmp/parson_TCPUDPout.csv TCPUDPout.csv
/bin/ln -s /home/kutztown.edu/parson/tmp/parson_Ethout.csv Ethout.csv
/bin/ln -s /home/kutztown.edu/parson/tmp/parson_TCPStreams.csv TCPStreams.csv
diff --strip-trailing-cr Ethout.csv /home/kutztown.edu/parson/DataMine/Ethout.csv.ref > Ethout.dif
 # The above diff comparison passes because I have supplied that part of the solution code.
diff --strip-trailing-cr TCPUDPout.csv /home/kutztown.edu/parson/DataMine/TCPUDPout.csv.ref > TCPUDPout.dif
diff: TCPUDPout.csv: No such file or directory
make: *** [test] Error 2

That diff test fails because your code has not yet run to create its output files. My working code creates Ethout.csv that records Ethernet datagrams from my home machine using Wireshark https://www.wireshark.org/. Your code will create TCPUDPout.csv that records individual TCP and UDP datagrams from the same input dataset. Your code will also create output file TCPStreams.csv that shows multiple-datagram TCP conversations between my client machine and assorted servers. My code serves as the project framework. My code that creates Ethout.csv serves as a template for the code you must write.

All project code requirements are documented with upper case **STUDENT** comments in csc523F23TCPUDP.py. When **make clean test** runs and you have re-checked all project requirements, use shell command **make turnitin** and follow the prompts to turn the assignment in to me by the deadline.

You can run **make unzip** on acad to extract
~parson/DataMine/WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt.gz
into WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt in your directory. It is big.

$ **make unzip**
/usr/bin/zcat /home/kutztown.edu/parson/DataMine/WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt.gz > WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt
$ **ls -l Wire***
-rw-r--r--. 1 parson domain users 72012365 Aug 30 16:29 WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt

You can run a local copy of your code without using the makefile like this, where python3 is your Python 3.x interpreter as documented at the end of this handout.

Here is the top of raw input data in the latter text file:
$ **head WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt**
No. Time Source Destination Protocol Length Info
1 0.000000 a104-126-112-105.deploy.static.akamaitechnologies.com 172.16.42.4 TCP 54 https(443) → 49243 [RST] Seq=1 Win=0 Len=0

Frame 1: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: 92:55:48:24:40:70 (92:55:48:24:40:70), Dst: Apple_e1:fa:60 (3c:15:c2:e1:fa:60)
Internet Protocol Version 4, Src: a104-126-112-105.deploy.static.akamaitechnologies.com (104.126.112.105), Dst: 172.16.42.4 (172.16.42.4)
Transmission Control Protocol, Src Port: https (443), Dst Port: 49243 (49243), Seq: 1, Len: 0

No. Time Source Destination Protocol Length Info
2 0.000003 a104-126-112-105.deploy.static.akamaitechnologies.com 172.16.42.4 TCP 54 https(443) → 49243 [RST] Seq=1 Win=0 Len=0

Here is the bottom:
$ **tail WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt**
Domain Name System (query)

No. Time Source Destination Protocol Length Info
109387 1259.096064 172.16.42.1 172.16.42.4 DNS 83 Standard query response 0x6100 No such name SRV _ldap._tcp.kutztown.edu

Frame 109387: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
Ethernet II, Src: 92:55:48:24:40:70 (92:55:48:24:40:70), Dst: Apple_e1:fa:60 (3c:15:c2:e1:fa:60)
Internet Protocol Version 4, Src: 172.16.42.1 (172.16.42.1), Dst: 172.16.42.4 (172.16.42.4)
User Datagram Protocol, Src Port: domain (53), Dst Port: 54157 (54157)
Domain Name System (response)

Here are some sample lines from the correct Ethout.csv output file:
$ **less ~parson/DataMine/Ethout.csv.ref**

inlinenum	frameNumber	timeStamp	frameBytes	frameProtocol	symsrc	numsrc	symdst	numdst
5	1	0	54	TCP	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60
13	2	0.000003	54	TCP	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60
21	3	0.009241	66	TCP	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60
29	4	0.010581	1514	TLSv1.2	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60
38	5	0.01085	1514	TCP	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60
9775	1149	8.13091	75	DNS	Apple_e1:fa:60	3c:15:c2:e1:fa:60	92:55:48:24:40:70	92:55:48:24:40:70
9784	1150	8.133043	203	DNS	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60
9793	1151	8.133749	78	TCP	Apple_e1:fa:60	3c:15:c2:e1:fa:60	92:55:48:24:40:70	92:55:48:24:40:70
9801	1152	8.138657	46	UDP	Apple_e1:fa:60	3c:15:c2:e1:fa:60	92:55:48:24:40:70	92:55:48:24:40:70
9812	1153	8.139736	70	ICMP	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60
9820	1154	8.162506	74	TCP	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60

The above CSV data fields (attributes) are defined as follows:

inlinenum	Line number in the input file.
frameNumber	Ethernet frame (datagram) number.
timestamp	Time of capture in fractional seconds.
frameBytes	Number of bytes in the Ethernet frame.
frameProtocol	Next level of protocol inside Ethernet frame.
symsrc	Symbolic Ethernet source (mac) address.
numsrc	Numeric Ethernet source (mac) address.
symdst,	Symbolic Ethernet destination (mac) address.
numdst	Numeric Ethernet destination (mac) address.

Here are some sample lines from the correct TCPUDPout.csv output file:
$ **less ~parson/DataMine/TCPUDPout.csv.ref**

inlinenum	frameNumber	timeStamp	ipSymSrc	ipNumSrc	ipSymDst	ipNumDst	protocol
831	98	1.236171	adobelogin.prod…	3.222.132.152	172.16.42.4	172.16.42.4	TCP
839	99	1.236213	172.16.42.4	172.16.42.4	adobelogin.prod…	3.222.132.152	TCP
847	100	1.236534	172.16.42.4	172.16.42.4	172.16.42.1	172.16.42.1	UDP
856	101	1.236622	172.16.42.4	172.16.42.4	172.16.42.1	172.16.42.1	UDP

portSymSrc	portNumSrc	portSymDst	portNumDst	tcpseq	tcplen	udptype
https	443	49249	49249	1449	1448	
49249	49249	https	443	518	0	
52937	52937	domain	53			Domain Name System (query)
52937	52937	domain	53			Domain Name System (query)

The above CSV data fields (attributes) are defined as follows:

inlinenum	Line number in the input file.
frameNumber	Ethernet frame (datagram) number.
timestamp	Time of capture in fractional seconds.
ipSymSrc	Symbolic Internet protocol source address.
ipNumSrc	Numeric Internet protocol source address.
ipSymDst	Symbolic Internet protocol destination address.
ipNumDst	Numeric Internet protocol destination address.
protocol	TCP or UDP in this assignment. The type of sub-protocol carried by IP.
portSymSrc	Symbolic TCP or UDP source port number.
portNumSrc	Numeric TCP or UDP source port number.
portSymDst	Symbolic TCP or UDP destination port number.
portNumDst	Numeric TCP or UDP destination port number.
tcpseq	TCP datagram sequence number, empty (Python None) for UDP.
tcplen	TCP datagram length, empty (Python None) for UDP.
udptype	Stripped line following “User Datagram Protocol” line in input file.

Here are some sample lines from the correct TCPStreams.csv output file:
$ **less ~parson/DataMine/TCPStreams.csv.ref**

minTimeStamp	maxTimeStamp	minFrameNum	maxFrameNum	minLineno	maxLineno	minSeq	maxSeq
0	0.000003	1	2	7	15	1	1
0.009241	0.067044	3	30	23	253	1	5181
0.017179	0.08956	10	33	84	277	0	3251
0.980936	1.111653	36	67	303	566	0	4243

client	cliDatagrams	cliBytes	server	svrDatagrams	svrBytes
104.126.112.105:443	2	0	172.16.42.4:49243	0	0
13.225.222.67:443	9	5211	172.16.42.4:49244	9	320
172.16.42.4:49245	7	300	17.253.15.202:80	6	3250
172.16.42.4:49246	15	1709	35.174.51.255:443	12	4241

The above CSV data fields (attributes) are defined as follows:

minTimeStamp	Initial time of capture of this TCP conversation in fractional seconds.
maxTimeStamp	Final time of capture of this TCP conversation in fractional seconds.
minFrameNum	Initial Ethernet frame number for this TCP conversation.
maxFrameNum	Final Ethernet frame number for this TCP conversation.
minLineno	Initial input file line number for this TCP conversation.

maxLineno	Final input file line number for this TCP conversation.
minSeq	Initial TCP sequence number for this TCP conversation.
maxSeq	Final TCP sequence number for this TCP conversation.
client	Client (originating) IP:PORT address.
cliDatagrams	Number of datagrams from client to server.
cliBytes	Number of bytes from client to server within those datagrams.
server	Server (initial receiving) IP:PORT address.
svrDatagrams	Number of datagrams from server to client.
svrBytes	Number of bytes from server to client within those datagrams.

Remember to double check the **STUDENT** specs in csc523F23TCPUDP.py. Run **make clean test** one last time after any changes, then **make turnitin** before the 11:59 PM deadline to avoid the 10% per day penalty.

Added notes on Ethernet matching re patterns supplied by Dr. Parson follow. From

```
# Ethernet II, Src: ADDR1A (ADDR1B), Dst: ADDR2A (ADDR2B)
#       where ADDR1A and ADDR2A *may be* symbolic, and the B's are like this:
# Ethernet II, Src: Apple_e1:fa:60 (3c:15:c2:e1:fa:60), Dst: 92:55:48:24:40:70 (92:55:48:24:40:70)
#   Note - values inside () are numeric, preceded by their symbolic values.
#   We need to record and report both, even when they are the same.

Ethernet_pattern_string = \
r'^Ethernet II, Src:\s+(\S+)\s+\(([^)]+)\),\s+Dst:\s+(\S+)\s+\(([^)]+)\)'
Ethernet_pattern = re.compile(Ethernet_pattern_string)
```

This pattern **(\S+)** or **(\S+)** matches any non-whitespace contiguous string such as **Apple_e1:fa:60** or **92:55:48:24:40:70**. The plus sign is inside the parentheses so that all matched characters are part of group(1) or group(3) in the handout code:

```
matchobj = Ethernet_pattern.match(inline)
if matchobj:
   demoEthernetWriter.writerow([lineno, frameNumber,
      "%.6f" % timeStamp, # format to 6 decimal places
      frameBytes, frameProtocol, matchobj.group(1),
      matchobj.group(2), matchobj.group(3), matchobj.group(4)])
   # Keep frameNumber & timeStamp for IP and TCP|UDP output.
   Continue
```

Here is the top of Ethout.csv:

inlinenum	frameNumber	timeStamp	frameBytes	frameProtocol	symsrc	numsrc	symdst	numdst
5	1	0	54	TCP	92:55:48:24:40:70	92:55:48:24:40:70	Apple_e1:fa:60	3c:15:c2:e1:fa:60

This pattern \(([^)]+)\) or \(([^)]+)\) matches the literal leading '(' and trailing ')' in the datagram, surrounding any non-) contiguous string, up to but not including the closing ')', such as **(3c:15:c2:e1:fa:60)** or **(92:55:48:24:40:70)**. The plus sign is inside the inner, grouping (non-escaped) parentheses so that all matched characters are part of group(2) or group(4) in the handout code

ADDED for running interactively at home. Use the interactive Terminal utility if you are on a Mac or the Windows CMD prompt terminal window.

$ **python csc523F23TCPUDP.py**
WireShark29Aug2023_FB_emails_WhyDidWaltKillGusCollapsed.txt.gz TCPUDPout.csv TCPStreams.csv > Ethout.csv
$ **python textdiff.py Ethout.csv Ethout.csv.ref**
$ **python textdiff.py TCPUDPout.csv TCPUDPout.csv.ref**
$ **python textdiff.py TCPStreams.csv TCPStreams.csv.ref**

If possible run one final **make clean test** and then **make turnitin** on acad or mcgonagall. Some students' Linux accounts on acad have not been set up as of the start of classes. If that is your situation, run the above **interactive commands** with a python version 3 in the last paragraph. Then turn in your source file via D2L Assignment 1 as follows.

Assignment 1 Python Regular expressions ONLY FOR STUDENTS WITH NO ACAD ACCOUNT

Due at the end of Thursday September 28. Turn in only csc523F23TCPUDP.py via D2L only if you cannot use acad.