In [ ]:

```
Agenda='''

File Handling

Exception Handling

How to use REGEX and TEXTFSM
Regular expressions
Pattern Writing
Compiling
Match/Search
Group/Groups
findall
re.sub
re.split

TEXTFSM Installation
Using textfsm

Coding Standards (PEP 8).

Self-documenting code (Pydoc).

Best practices of programming;
Writing efficient code

Virtual environment

Unit Testing
Testing Fundamental
Types of Testing
Unittest Framework
Run Test
doctest
Write Unittest.TestCase for Python Code

HTTP/HTTPS calls – What it is used for?
What is certification/authentication
How to use HTTP library in python; examples of HTTP/HTTPS endpoint calls.
Encoding/decoding formats

JSON, XML – Overview

Uses of JSON / XML using python

Serialisation
YAML / JINJA– What/how/why do we use.

Virtual environment
'''
```

In [ ]:

```
print('''File Handling
1. Create File
2. Write Content Into file
3. Read Content of the File
4. Append Content
5. Various mode ( r, r+, w, w+, a, a+)
6. Insert Content into file
7. Use of with statement
''')

```

In [ ]:

```
# Create agenda4.txt File

file1 = open('agenda4.txt','w+')

# Write Content into file

file1.write(Agenda)

# Close file

file1.close()

# Open File for reading ( default is read operation)

file1 = open('agenda4.txt')

# Read content of the file

print(file1.read())

print("\n"*2)

# Close file

file1.close()


# Create agenda4.txt File

file1 = open('agenda4.txt','a+')

# Write Content into file

file1.write("New Agenda Appended \n")

# Close file

file1.close()

# Open File for reading after append.

file1 = open('agenda4.txt')

# Read content of the file

print(file1.read())

# Close file

file1.close()
```

In [ ]:

```
# Create agenda5.txt File using with

with open('agenda5.txt','w+') as file1:
    file1.write(Agenda)


# Open File for reading ( default is read operation)

with open('agenda5.txt') as file1:
    print(file1.read())

print("\n"*2)


# Create agenda5.txt with append mode File

with open('agenda5.txt','a+') as file1:
    file1.write("New Agenda Appended \n")


# Open File for reading after append.

with open('agenda5.txt') as file1:
    print(file1.read())
```

In [ ]:

```
data = int(input("Enter some data:"))
```

In [ ]:

```
try:
    data = int(input("Enter some data:"))
except ValueError as e:
    print("Write Integer Data Only::", e)
else:
    print("No Exception. Data is %d "%data)

```

In [ ]:

```
while True:
    try:
        data = int(input("Enter some data:"))
    except ValueError:
        print("Write Integer Data Only")
    else:
        print("No Exception. Data is %d "%data)
        break

```

In [ ]:

```
while True:
    try:
        data = int(input("Enter some data:"))
    except ValueError:
        print("Write Integer Data Only")
    except Exception:
        print("Other Exception")
    else:
        print("No Exception. Data is %d "%data)
        break
    finally:
        print("It is always executed\n")


```

In [ ]:

```
help(ValueError)
```

In [ ]:

```
help(Exception)
```

In [ ]:

```
# User Define Exception

class ShortInputException(Exception):
    '''Short Input Exception class for
    short message. Expected minimum 6
    characters length'''

    def __init__(self, length, atleast):
        ''' Intialize ShortInputException Class'''
        self.length = length  # Object Variable
        self.atleast = atleast

        print("You Message size is %d bytes,\
 Expected %d bytes"%(self.length,self.atleast))


```

In [ ]:

```
Obj1 = ShortInputException(5,6)
```

In [ ]:

```
while True:
    try:
        data = input("Enter some data:")
        if len(data) < 6:
            raise ShortInputException(len(data),6)
    except ShortInputException as e:
        print(e)
    except Exception as e1:
        print("Other Exception",e1)
    else:
        print("Data is %s "%data)
        break
    finally:
        print("It is always executed\n")
```

In [ ]:

```
import re

# Create Pattern Object using Compile Method

pattern = re.compile('[a-z]+')

print(pattern)

# match data with pattern

match1 = pattern.match('surendra panpaliya')

# Check Match status and print if it occurs

if match1:
    print("Match Occurred::",match1.group())
    print("Match Start and End::",match1.span())
else:
    print("Match Not Occurred")
```

In [ ]:

```
print(dir(match1))
```

In [ ]:

```
import re

# Create Pattern Object using Compile Method

pattern = re.compile('[a-z]+')

print(pattern)

# match data with pattern

match1 = pattern.search('876876 surendra panpaliya')

# Check Match status and print if it occurs

if match1:
    print("Match Occurred::",match1.group())
    print("Match Start and End::",match1.span())
else:
    print("Match Not Occurred")
```

In [13]:

```
data = '''

File Handling
Email1 : surendra.panpaliya@gmail.com

Mobile Number1 : 9975072320

Account Number1 : 624001053983

Exception Handling

Email1 : surendra@gktcs.com

How to use REGEX and TEXTFSM

Account Number2 : 624001053989

Mobile Number2 : 9975072321

Regular expressions
Pattern Writing
Compiling

Mobile Number : 9975072322

Match/Search

Account Number3 : 624001053984

Group/Groups
Email1 : surendra@gktcs.co.in

findall
re.sub
re.split

TEXTFSM Installation
Using textfsm

Coding Standards (PEP 8).

Self-documenting code (Pydoc).

Best practices of programming;
Writing efficient code

'''

# Create agenda5.txt File using with

with open('adata.txt','w+') as file1:
    file1.write(data)


# Open File for reading ( default is read operation)

with open('adata.txt') as file1:
    print(file1.read())
```

File Handling
Email1 : surendra.panpaliya@gmail.com

Mobile Number1 : 9975072320

Account Number1 : 624001053983

Exception Handling

Email1 : surendra@gktcs.com

How to use REGEX and TEXTFSM

Account Number2 : 624001053989

Mobile Number2 : 9975072321

Regular expressions
Pattern Writing
Compiling

Mobile Number : 9975072322

Match/Search

Account Number3 : 624001053984
Group/Groups
Email1 : surendra@gktcs.co.in

findall
re.sub
re.split

TEXTFSM Installation
Using textfsm

Coding Standards (PEP 8).

Self-documenting code (Pydoc).

Best practices of programming;
Writing efficient code

In [14]:

```
# Extract Mobile Number from data

Mobiles = re.findall(r'\d{10}',data)

# \d match 10 digit

print(Mobiles)
```

```
['9975072320', '6240010539', '6240010539', '9975072321', '9975072322
', '6240010539']
```

In [18]:

```
# Extract Mobile Number from data

Mobiles = re.findall(r'\b\d{10}\b',data)

# \b to match boundry data

print(Mobiles)

# Substitute / replace Mobile Number of data with MNUMxxx

Substitute_Mobile = re.sub(r'\b\d{10}\b','MNUMxxx',data,2)

print(Substitute_Mobile)
```

```
['9975072320', '9975072321', '9975072322']


File Handling
Email1 : surendra.panpaliya@gmail.com

Mobile Number1 : MNUMxxx

Account Number1 : 624001053983

Exception Handling

Email1 : surendra@gktcs.com

How to use REGEX and TEXTFSM

Account Number2 : 624001053989

Mobile Number2 : MNUMxxx

Regular expressions
Pattern Writing
Compiling
```

Mobile Number : 9975072322

Match/Search

Account Number3 : 624001053984
Group/Groups
Email1 : surendra@gktcs.co.in

findall
re.sub
re.split

TEXTFSM Installation
Using textfsm

Coding Standards (PEP 8).

Self-documenting code (Pydoc).

Best practices of programming;
Writing efficient code

In [16]:

```
# Extract Account Number from data

Account_Numbers = re.findall(r'\b\d{12}\b',data)

# \b to match boundry data

print(Account_Numbers)
```

```
['624001053983', '624001053989', '624001053984']
```

In [ ]:

```
# Extract Email Address from data

Email_Address = re.findall(r'\b[.\w]+@\w+.[.\w]{2,5}\b',data)

# \b to match boundry data
# \w alphanumeric character ( a-z, A-Z and 0-9)
# [. \w] means . and alphanumeric are optional
# [.\w]+ One or More Occurance of alphanumeric and '.'
# [.\w]{2,5} match 2 to 5 occurrences of alphanumeric

print(Email_Address)
```

In [ ]:

```
import fileinput
import re
import glob


# search for text file using glob and create list of text file

list_of_files = glob.glob("ad*.txt")

print(list_of_files)

# Read all listed files using fileinput module

read_files = fileinput.input(list_of_files)

for line in read_files:
    print(line.strip())
```

In [3]:

```
import fileinput
import re
import glob


# search for text file using glob and create list of text file

list_of_files = glob.glob("ad*.txt")

print(list_of_files)

# Read all listed files using fileinput module
with fileinput.input(list_of_files) as read_files:
    pattern = re.compile(r'\b[.\w]+@\w+.[.\w]{2,5}\b')
    for line in read_files:
        if pattern.search(line):
            print(read_files.filename(),read_files.lineno(),\
                  line.strip())
```

```
['adata.txt']
adata.txt 4 Email1 : surendra.panpaliya@gmail.com
adata.txt 12 Email1 : surendra@gktcs.com
adata.txt 30 Email1 : surendra@gktcs.co.in
```

In [ ]:

```
pattern = re.compile(r'\b[.\w]+@\w+.[.\w]{2,5}\b')

pattern.findall(data)
```

In [12]:

```
import fileinput
import re
import glob


# search for text file using glob and create list of text file

list_of_files = glob.glob("ad*.txt")

print(list_of_files)

# Read all listed files using fileinput module
with fileinput.input(list_of_files) as read_files:
    pattern = re.compile(r'\b[.\w]+@\w+.[.\w]{2,5}\b')
    for line in read_files:
        if pattern.findall(line):
            print(read_files.filename(),read_files.lineno(),\
                  line.strip())
```

```
['adata.txt']
adata.txt 4 Email1 : surendra.panpaliya@gmail.com
adata.txt 12 Email1 : surendra@gktcs.com
adata.txt 30 Email1 : surendra@gktcs.co.in
```

TextFSM

Textfsm is a text parsing library written in python to turn plain text into structured data. Originally created by Google, the project seemed largely abandoned until recently being added to github and receiving a small update. Let's extract interesting data from the output of the show interface command on a Cisco ASA using TextFSM. For reference the software versions used in this post are below. TextFSM - 0.3.2 Installation Create a virtual environment and install textfsm. virtualenv -p python2.7 ~/envs/textfsm-env source ~/envs/textfsm-env/bin/activate inside virtual environment pip install textfsm

In [19]:

```
'''Usage
The good folks over at network.toCode() maintain a library of templates and

Use curl to download the cisco_asa_show_interface.template template.

```

```
curl -O https://raw.githubusercontent.com/networktocode/ntc-templates/master

Templates use a series of regular expressions to define the data to be extra
There is a pretty good explaination of the components of TextFSM here.

# textfsm template
Value Required INTERFACE (\S+)
Value INTERFACE_ZONE (.+?)
Value LINK_STATUS (\w+)
Value PROTOCOL_STATUS (.*)
Value HARDWARE_TYPE ([\w ]+)
Value BANDWIDTH (\d+\s+\w+)
Value DELAY (\d+\s+\w+)
Value DUPLEX (\w+\-\w+)
Value SPEED (\d+\w+\s\w+)
Value DESCRIPTION (.*)
Value ADDRESS ([a-zA-Z0-9]+.[a-zA-Z0-9]+.[a-zA-Z0-9]+)
Value MTU (\d+)
Value IP_ADDRESS (\d+\.\d+\.\d+\.\d+)
Value NET_MASK (\d+\.\d+\.\d+\.\d+)
Value ONEMIN_IN_PPS (\d+)
Value ONEMIN_IN_RATE (\d+)
Value ONEMIN_OUT_PPS (\d+)
Value ONEMIN_OUT_RATE (\d+)
Value ONEMIN_DROP_RATE (\d+)
Value FIVEMIN_IN_PPS (\d+)
Value FIVEMIN_IN_RATE (\d+)
Value FIVEMIN_OUT_PPS (\d+)
Value FIVEMIN_OUT_RATE (\d+)
Value FIVEMIN_DROP_RATE (\d+)

Start
  ^.*Interface ${INTERFACE} "${INTERFACE_ZONE}", is ${LINK_STATUS}.*protocol
  ^\s+Hardware is ${HARDWARE_TYPE} -> Continue
  ^.*BW ${BANDWIDTH}.*DLY ${DELAY}
  ^.*\(${DUPLEX}.*Auto-Speed\(${SPEED}\)
  ^.*Description: ${DESCRIPTION}
  ^.*MAC address ${ADDRESS}.*MTU ${MTU}
  ^.*IP address ${IP_ADDRESS}, .*subnet mask ${NET_MASK}
  ^.*1 minute input rate ${ONEMIN_IN_PPS} pkts/sec,\s+${ONEMIN_IN_RATE} byte
  ^.*1 minute output rate ${ONEMIN_OUT_PPS} pkts/sec,\s+${ONEMIN_OUT_RATE} b
  ^.*1 minute drop rate, ${ONEMIN_DROP_RATE}
  ^.*5 minute input rate ${FIVEMIN_IN_PPS} pkts/sec,\s+${FIVEMIN_IN_RATE} by
  ^.*5 minute output rate ${FIVEMIN_OUT_PPS} pkts/sec,\s+${FIVEMIN_OUT_RATE}
  ^.*5 minute drop rate, ${FIVEMIN_DROP_RATE} -> Record  '''

#Lets fire up a python interpreter and create an interfaces variable with th
#command from a Cisco ASA.
# output of show interface
interfaces = '''
Interface GigabitEthernet0/0 "inside", is up, line protocol is up
  Hardware is i82540EM rev02, BW 1000 Mbps, DLY 10 usec
    Auto-Duplex(Full-duplex), Auto-Speed(1000 Mbps)
    Input flow control is unsupported, output flow control is off
    MAC address 0800.2735.03c6, MTU 1500
    IP address 169.254.1.11, subnet mask 255.255.255.0
    0 packets input, 0 bytes, 0 no buffer
    Received 0 broadcasts, 0 runts, 0 giants
```

```
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
    0 pause input, 0 resume input
    0 L2 decode drops
    1 packets output, 60 bytes, 0 underruns
    0 pause output, 0 resume output
    0 output errors, 0 collisions, 1 interface resets
    0 late collisions, 0 deferred
    0 input reset drops, 0 output reset drops
    input queue (blocks free curr/low): hardware (511/511)
    output queue (blocks free curr/low): hardware (511/510)
  Traffic Statistics for "inside":
    0 packets input, 0 bytes
    1 packets output, 28 bytes
    0 packets dropped
      1 minute input rate 0 pkts/sec,  0 bytes/sec
      1 minute output rate 0 pkts/sec,  0 bytes/sec
      1 minute drop rate, 0 pkts/sec
      5 minute input rate 0 pkts/sec,  0 bytes/sec
      5 minute output rate 0 pkts/sec,  0 bytes/sec
      5 minute drop rate, 0 pkts/sec
Interface Management0/0 "management", is up, line protocol is up
  Hardware is i82540EM rev02, BW 1000 Mbps, DLY 10 usec
    Auto-Duplex(Full-duplex), Auto-Speed(1000 Mbps)
    Input flow control is unsupported, output flow control is off
    MAC address 0800.277d.ea42, MTU 1500
    IP address 10.0.2.15, subnet mask 255.255.255.0
    1086 packets input, 83965 bytes, 0 no buffer
    Received 2 broadcasts, 0 runts, 0 giants
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
    0 pause input, 0 resume input
    0 L2 decode drops
    1071 packets output, 130571 bytes, 0 underruns
    0 pause output, 0 resume output
    0 output errors, 0 collisions, 1 interface resets
    0 late collisions, 0 deferred
    0 input reset drops, 0 output reset drops
    input queue (blocks free curr/low): hardware (497/458)
    output queue (blocks free curr/low): hardware (510/506)
  Traffic Statistics for "management":
        1085 packets input, 62837 bytes
    1071 packets output, 114891 bytes
    2 packets dropped
      1 minute input rate 16 pkts/sec,  905 bytes/sec
      1 minute output rate 16 pkts/sec,  1762 bytes/sec
      1 minute drop rate, 0 pkts/sec
      5 minute input rate 0 pkts/sec,  0 bytes/sec
      5 minute output rate 0 pkts/sec,  0 bytes/sec
      5 minute drop rate, 0 pkts/sec
'''
```

In [22]:

```
#Next up we will open the cisco_asa_show_interface.template
#file and run the interfaces data through the parser.

# import library
import textfsm

#print(help(textfsm))
print("\n\n")
# open the template file
with open('cisco_asa_show_interface.template', 'r') as f:
    template = textfsm.TextFSM(f)

# run the interface data through the template parser
print(template.ParseText(interfaces))


```

```
[['GigabitEthernet0/0', 'inside', 'up', 'up', 'i82540EM rev02', '100
0 Mbps', '10 usec', 'Full-duplex', '1000 Mbps', '', '0800.2735.03c6'
, '1500', '169.254.1.11', '255.255.255.0', '0', '0', '0', '0', '0',
'0', '0', '0', '0', '0'], ['Management0/0', 'management', 'up', 'up'
, 'i82540EM rev02', '1000 Mbps', '10 usec', 'Full-duplex', '1000 Mbp
s', '', '0800.277d.ea42', '1500', '10.0.2.15', '255.255.255.0', '16'
, '905', '16', '1762', '0', '0', '0', '0', '0', '0']]
```

In [23]:

```
1 cat 'cisco_asa_show_interface.template'
```

```
Value Required INTERFACE (\S+)
Value INTERFACE_ZONE (.+?)
Value LINK_STATUS (\w+)
Value PROTOCOL_STATUS (.*)
Value HARDWARE_TYPE ([\w ]+)
Value BANDWIDTH (\d+\s+\w+)
Value DELAY (\d+\s+\w+)
Value DUPLEX (\w+\-\w+)
Value SPEED (\d+\w+\s\w+)
Value DESCRIPTION (.*)
Value ADDRESS ([a-zA-Z0-9]+.[a-zA-Z0-9]+.[a-zA-Z0-9]+)
Value MTU (\d+)
Value IP_ADDRESS (\d+\.\d+\.\d+\.\d+)
Value NET_MASK (\d+\.\d+\.\d+\.\d+)
Value ONEMIN_IN_PPS (\d+)
Value ONEMIN_IN_RATE (\d+)
Value ONEMIN_OUT_PPS (\d+)
Value ONEMIN_OUT_RATE (\d+)
Value ONEMIN_DROP_RATE (\d+)
Value FIVEMIN_IN_PPS (\d+)
Value FIVEMIN_IN_RATE (\d+)
Value FIVEMIN_OUT_PPS (\d+)
Value FIVEMIN_OUT_RATE (\d+)
Value FIVEMIN_DROP_RATE (\d+)

Start
  ^Interface.* -> Continue.Record
  ^.*Interface ${INTERFACE} "${INTERFACE_ZONE}", is ${LINK_STATUS}.*
protocol is ${PROTOCOL_STATUS}
  ^\s+Hardware is ${HARDWARE_TYPE} -> Continue
  ^.*BW ${BANDWIDTH}.*DLY ${DELAY}
  ^.*\(${DUPLEX}.*Auto-Speed\(${SPEED}\)
  ^.*Description: ${DESCRIPTION}
  ^.*MAC address ${ADDRESS}.*MTU ${MTU}
  ^.*IP address ${IP_ADDRESS}, .*subnet mask ${NET_MASK}
  ^.*1 minute input rate ${ONEMIN_IN_PPS} pkts/sec,\s+${ONEMIN_IN_RA
TE} bytes/sec
  ^.*1 minute output rate ${ONEMIN_OUT_PPS} pkts/sec,\s+${ONEMIN_OUT
_RATE} bytes/sec
  ^.*1 minute drop rate, ${ONEMIN_DROP_RATE}
  ^.*5 minute input rate ${FIVEMIN_IN_PPS} pkts/sec,\s+${FIVEMIN_IN_
RATE} bytes/sec
  ^.*5 minute output rate ${FIVEMIN_OUT_PPS} pkts/sec,\s+${FIVEMIN_O
UT_RATE} bytes/sec
  ^.*5 minute drop rate, ${FIVEMIN_DROP_RATE}
```

In [24]:

```
cisco_asa_data = '''
Value Required INTERFACE (\S+)
Value INTERFACE_ZONE (.+?)
Value LINK_STATUS (\w+)
Value PROTOCOL_STATUS (.*)
Value HARDWARE_TYPE ([\w ]+)
Value BANDWIDTH (\d+\s+\w+)
Value DELAY (\d+\s+\w+)
Value DUPLEX (\w+\-\w+)
Value SPEED (\d+\w+\s\w+)
Value DESCRIPTION (.*)
Value ADDRESS ([a-zA-Z0-9]+.[a-zA-Z0-9]+.[a-zA-Z0-9]+)
Value MTU (\d+)
Value IP_ADDRESS (\d+\.\d+\.\d+\.\d+)
Value NET_MASK (\d+\.\d+\.\d+\.\d+)
Value ONEMIN_IN_PPS (\d+)
Value ONEMIN_IN_RATE (\d+)
Value ONEMIN_OUT_PPS (\d+)
Value ONEMIN_OUT_RATE (\d+)
Value ONEMIN_DROP_RATE (\d+)
Value FIVEMIN_IN_PPS (\d+)
Value FIVEMIN_IN_RATE (\d+)
Value FIVEMIN_OUT_PPS (\d+)
Value FIVEMIN_OUT_RATE (\d+)
Value FIVEMIN_DROP_RATE (\d+)

Start
  ^Interface.* -> Continue.Record
  ^.*Interface ${INTERFACE} "${INTERFACE_ZONE}", is ${LINK_STATUS}.*protocol
  ^\s+Hardware is ${HARDWARE_TYPE} -> Continue
  ^.*BW ${BANDWIDTH}.*DLY ${DELAY}
  ^.*\(${DUPLEX}.*Auto-Speed\(${SPEED}\)
  ^.*Description: ${DESCRIPTION}
  ^.*MAC address ${ADDRESS}.*MTU ${MTU}
  ^.*IP address ${IP_ADDRESS}, .*subnet mask ${NET_MASK}
  ^.*1 minute input rate ${ONEMIN_IN_PPS} pkts/sec,\s+${ONEMIN_IN_RATE} bytes
  ^.*1 minute output rate ${ONEMIN_OUT_PPS} pkts/sec,\s+${ONEMIN_OUT_RATE} by
  ^.*1 minute drop rate, ${ONEMIN_DROP_RATE}
  ^.*5 minute input rate ${FIVEMIN_IN_PPS} pkts/sec,\s+${FIVEMIN_IN_RATE} byt
  ^.*5 minute output rate ${FIVEMIN_OUT_PPS} pkts/sec,\s+${FIVEMIN_OUT_RATE}
  ^.*5 minute drop rate, ${FIVEMIN_DROP_RATE}
'''



# Create 'cisco_asa_show_interface1.template' File using with

with open('cisco_asa_show_interface1.template','w+') as file1:
    file1.write(cisco_asa_data)


# Open File for reading ( default is read operation)

with open('cisco_asa_show_interface1.template') as file1:
    print(file1.read())
```

```
Value Required INTERFACE (\S+)
```

```
Value INTERFACE_ZONE (.+?)
Value LINK_STATUS (\w+)
Value PROTOCOL_STATUS (.*)
Value HARDWARE_TYPE ([\w ]+)
Value BANDWIDTH (\d+\s+\w+)
Value DELAY (\d+\s+\w+)
Value DUPLEX (\w+\-\w+)
Value SPEED (\d+\w+\s\w+)
Value DESCRIPTION (.*)
Value ADDRESS ([a-zA-Z0-9]+.[a-zA-Z0-9]+.[a-zA-Z0-9]+)
Value MTU (\d+)
Value IP_ADDRESS (\d+\.\d+\.\d+\.\d+)
Value NET_MASK (\d+\.\d+\.\d+\.\d+)
Value ONEMIN_IN_PPS (\d+)
Value ONEMIN_IN_RATE (\d+)
Value ONEMIN_OUT_PPS (\d+)
Value ONEMIN_OUT_RATE (\d+)
Value ONEMIN_DROP_RATE (\d+)
Value FIVEMIN_IN_PPS (\d+)
Value FIVEMIN_IN_RATE (\d+)
Value FIVEMIN_OUT_PPS (\d+)
Value FIVEMIN_OUT_RATE (\d+)
Value FIVEMIN_DROP_RATE (\d+)

Start
  ^Interface.* -> Continue.Record
  ^.*Interface ${INTERFACE} "${INTERFACE_ZONE}", is ${LINK_STATUS}.*
protocol is ${PROTOCOL_STATUS}
  ^\s+Hardware is ${HARDWARE_TYPE} -> Continue
  ^.*BW ${BANDWIDTH}.*DLY ${DELAY}
  ^.*\(${DUPLEX}.*Auto-Speed\(${SPEED}\)
  ^.*Description: ${DESCRIPTION}
  ^.*MAC address ${ADDRESS}.*MTU ${MTU}
  ^.*IP address ${IP_ADDRESS}, .*subnet mask ${NET_MASK}
  ^.*1 minute input rate ${ONEMIN_IN_PPS} pkts/sec,\s+${ONEMIN_IN_RA
TE} bytes/sec
  ^.*1 minute output rate ${ONEMIN_OUT_PPS} pkts/sec,\s+${ONEMIN_OUT
_RATE} bytes/sec
  ^.*1 minute drop rate, ${ONEMIN_DROP_RATE}
  ^.*5 minute input rate ${FIVEMIN_IN_PPS} pkts/sec,\s+${FIVEMIN_IN_
RATE} bytes/sec
  ^.*5 minute output rate ${FIVEMIN_OUT_PPS} pkts/sec,\s+${FIVEMIN_O
UT_RATE} bytes/sec
  ^.*5 minute drop rate, ${FIVEMIN_DROP_RATE}
```

In [25]:

```
1 import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it
.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

In [39]:

```
%%writefile country_data.xml
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
    <country name="India">
        <rank>5</rank>
        <year>2018</year>
        <gdppc>136</gdppc>
        <neighbor name="Singapoor" direction="W"/>
        <neighbor name="Malasiya" direction="E"/>
    </country>
</data>
```

```
Overwriting country_data.xml
```

In [79]:

```
import xml.etree.ElementTree as ET
tree = ET.parse("country_data.xml")
root = tree.getroot()
print(root.tag)
```

```
data
```

In [36]:

```
#As an Element, root has a tag and a dictionary of attributes:
print(root.tag)

print(root.attrib)
#It also has children nodes over which we can iterate:

for child in root:
    print(child.tag, child.attrib)
```

```
data
{}
country {'name': 'Liechtenstein'}
country {'name': 'Singapore'}
country {'name': 'Panama'}
country {'name': 'India'}
```

In [74]:

```
print(root.tag)
print(root[0].tag)
print("root county1 child tag is %s and value is %s"\
      %(root[0][0].tag,root[0][0].text))
print("root county1 child tag is %s and value is %s"\
      %(root[0][1].tag,root[0][1].text))
print("root county1 child tag is %s and value is %s"\
      %(root[0][2].tag,root[0][2].text))

print("root county1 child tag is %s and value is %s"\
      %(root[0][3].tag,root[0][3].attrib))

print("\n")

print("root county2 child tag is %s and value is %s"\
      %(root[1][0].tag,root[1][0].text))
print("root county2 child tag is %s and value is %s"\
      %(root[1][1].tag,root[1][1].text))
print("root county2 child tag is %s and value is %s"\
      %(root[1][2].tag,root[1][2].text))

print("root county2 child tag is %s and value is %s"\
      %(root[1][3].tag,root[1][3].attrib))


#print(root[0][3].attrib)
```

```
data
country
root county1 child tag is rank and value is 1
root county1 child tag is year and value is 2008
root county1 child tag is gdppc and value is 141100
root county1 child tag is neighbor and value is {'name': 'Austria',
'direction': 'E'}


root county2 child tag is rank and value is 4
root county2 child tag is year and value is 2011
root county2 child tag is gdppc and value is 59900
root county2 child tag is neighbor and value is {'name': 'Malaysia',
'direction': 'N'}
```

In [38]:

```
for neighbor in root.iter('neighbor'):
        print(neighbor.attrib)
```

```
{'name': 'Austria', 'direction': 'E'}
{'name': 'Switzerland', 'direction': 'W'}
{'name': 'Malaysia', 'direction': 'N'}
{'name': 'Costa Rica', 'direction': 'W'}
{'name': 'Colombia', 'direction': 'E'}
{'name': 'SingaPoor', 'direction': 'W'}
{'name': 'Malesiya', 'direction': 'E'}
```

In [75]:

```
for country in root.findall('country'):
    rank = country.find('rank').text
    name = country.get('name')
    print(name,rank)
```

```
Liechtenstein 1
Singapore 4
Panama 68
India 5
```

In [84]:

```
import xml.etree.ElementTree as ET
tree = ET.parse("country_data.xml")
root = tree.getroot()
print(root.tag)

for rank in root.iter('rank'):
    new_rank = int(rank.text) + 1
    rank.text = str(new_rank)
    print(rank.tag, rank.text)
    rank.set('updated', 'yes')

tree.write('country_updated.xml')
```

```
data
rank 2
rank 5
rank 69
rank 6
```

In [ ]:

```
# %load country_updated.xml
<data>
    <country name="Liechtenstein">
        <rank updated="yes">2</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor direction="E" name="Austria" />
        <neighbor direction="W" name="Switzerland" />
    </country>
    <country name="Singapore">
        <rank updated="yes">5</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor direction="N" name="Malaysia" />
    </country>
    <country name="Panama">
        <rank updated="yes">69</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor direction="W" name="Costa Rica" />
        <neighbor direction="E" name="Colombia" />
    </country>
    <country name="India">
        <rank updated="yes">6</rank>
        <year>2018</year>
        <gdppc>136</gdppc>
        <neighbor direction="W" name="Singapoor" />
        <neighbor direction="E" name="Malasiya" />
    </country>
</data>
```

In [ ]:

```

```