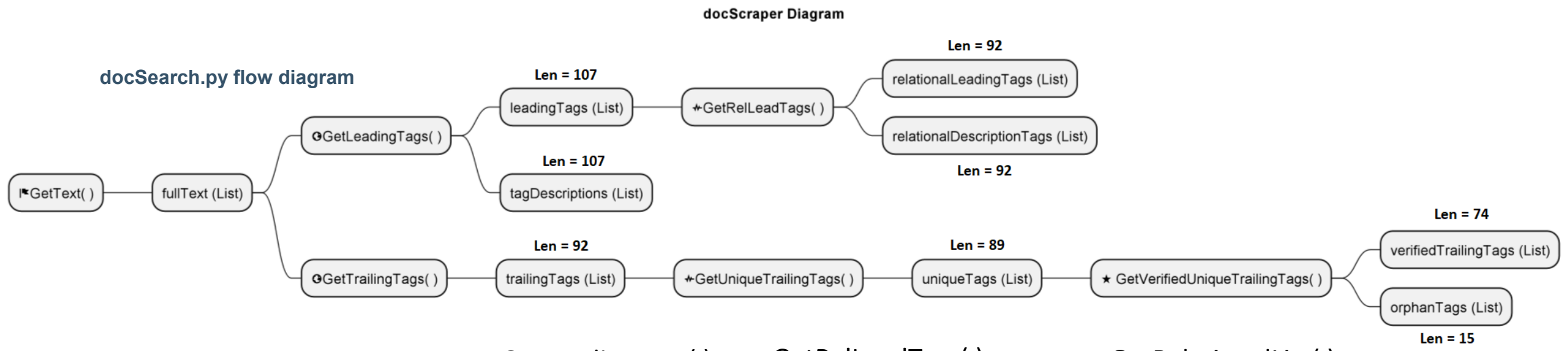


## GetText( )

Opens a document and extracts each line of the text as an element in a list.

## GetLeadingTags( )

Uses **GetText()** to extract all leading tags (using regular expressions) and leading tags with descriptions and trailing tags as an output of two lists.

## GetRelLeadTags( )

Uses **GetLeadingTags()** and eliminates all tags (**RISK** and **URS**) that do not have both leading and trailing tags, and outputs for both previous lists as a reduced set.

## GetRelationalList( )

Uses **GetVerifiedUniqueTrailingTags()** **GetTrailingTags( )**, and **GetLeadingTags**.
The verifiedTrailingTags list is used to iterate through it to find the indices in leadingTags list. Once the indices are found, it uses the relationalDescriptionsTags to build the output document.

**docExtract.py description**

## GetTrailingTags( )

Uses **GetText()** to extract all trailing tags as a list (using regular expressions).

## GetUniqueTrailingTags( )

Uses **GetTrailingTags()** to extract unique trailing tags and returns a sorted list in alphabetical order. It also eliminates unneeded tags like PASS/FAIL and N/A

## GetVerifiedUniqueTrailingTags( )

Uses **GetUniqueTrailingTags()** to extract tags that can be used to search through trailing tags along with an orphanTags list as an output.

Dictionary containing all tags and documents where they can be found
This section will be replaced by a function that reads information in from an external file.

```
import docx
import re

docFile = {"HRD":"HDS_new_pump.docx", "HRS":"HRS_new_pump.docx", "HTP":"HTP_new_pump.docx", "HTR":"HTR_new_pump.docx", \
           "PRS":"PRS_new_pump.docx", "RISK":"RiskAnalysis_Pump.docx", "SDS":"SDS_New_pump_x04.docx", \
           "ACE":"SRS_ACE_Pump_X01.docx", "BOLUS":"SRS_BolusCalc_Pump_X04.docx", "SRS":"SRS_DosingAlgorithm_X03.docx", \
           "SVAL":"SVaP_new_pump.docx", "SVATR":"SVaTR_new_pump.docx", "UT":"SVeTR_new_pump.docx", "URS":"URS_new_pump.docx"}

filePath = "C:/Users/steph/OneDrive/Desktop/Docs_Project/"

docFileList = list(docFile.keys())          # This is a list of all main tags found in each document
leadingTags = []                            # List of Leading Tags
trailingTags = []                           # List of Trailing Tags
relationalLeadingTags = []                  # List of Leading Tags excluding "RISK" and "URS" tags
tagDescriptions = []                        # List of all trags with descriptions
uniqueParentTags = []                       # Unique Trailing Tags List

def GetText(filename):                      # Opens the document and places each paragraph into a list
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    fullText = [ele for ele in fullText if ele.strip()]    # Eliminates empty paragraphs or spaces
    return fullText                         # List of all document content
```

# Dictionary Replacement code

```
docFile = {"HRD":"HDS_new_pump.docx", "HRS":"HRS_new_pump.docx", "HTP":"HTP_new_pump.docx", "HTR":"HTR_new_pump.docx", \
           "PRS":"PRS_new_pump.docx", "RISK":"RiskAnalysis_Pump.docx", "SDS":"SDS_New_pump_x04.docx", \
           "ACE":"SRS_ACE_Pump_X01.docx", "BOLUS":"SRS_BolusCalc_Pump_X04.docx", "SRS":"SRS_DosingAlgorithm_X03.docx", \
           "SVAL":"SVaP_new_pump.docx", "SVATR":"SVaTR_new_pump.docx", "UT":"SVeTR_new_pump.docx", "URS":"URS_new_pump.docx"}
```

The dictionary above is replaced by the code snippet below that reads values from a text file

```
docFile = {}

f = open("DocTagsList.txt")
for line in f:
    line = line.split()
    docFile[line[0]] = line[1]
```

Splits each line into a list and uses the first index as a key and the second index as a value for the dictionary docFile

## DocTagsList.txt content

```
HRD HDS_new_pump.docx
HRS HRS_new_pump.docx
HTP HTP_new_pump.docx
HTR HTR_new_pump.docx
PRS PRS_new_pump.docx
RISK RiskAnalysis_Pump.docx
SDS SDS_New_pump_x04.docx
ACE SRS_ACE_Pump_X01.docx
BOLUS SRS_BolusCalc_Pump_X04.docx
SRS SRS_DosingAlgorithm_X03.docx
SVAL SVaP_new_pump.docx
SVATR SVaTR_new_pump.docx
UT SVeTR_new_pump.docx
URS URS_new_pump.docx
```

```
import docx
import re

docFile = {"HRD":"HDS_new_pump.docx", "HRS":"HRS_new_pump.docx", "HTP":"HTP_new_pump.docx", "HTR":"HTR_new_pump.docx", \
           "PRS":"PRS_new_pump.docx", "RISK":"RiskAnalysis_Pump.docx", "SDS":"SDS_New_pump_x04.docx", \
           "ACE":"SRS_ACE_Pump_X01.docx", "BOLUS":"SRS_BolusCalc_Pump_X04.docx", "SRS":"SRS_DosingAlgorithm_X03.docx", \
           "SVAL":"SVaP_new_pump.docx", "SVATR":"SVaTR_new_pump.docx", "UT":"SVeTR_new_pump.docx", "URS":"URS_new_pump.docx"}

filePath = "C:/Users/steph/OneDrive/Desktop/Docs_Project/"
```

Creates a list of all tags found in the docFile dictionary used to verify valid tags

```
docFileList = list(docFile.keys())          # This is a list of all main tags found in each document
leadingTags = []                            # List of Leading Tags
trailingTags = []                           # List of Trailing Tags
relationalLeadingTags = []                  # List of Leading Tags excluding "RISK" and "URS" tags
tagDescriptions = []                        # List of all trags with descriptions
uniqueParentTags = []                       # Unique Trailing Tags List

def GetText(filename):                      # Opens the document and places each paragraph into a list
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    fullText = [ele for ele in fullText if ele.strip()]    # Eliminates empty paragraphs or spaces
    return fullText                         # List of all document content
```

```
import docx
import re

docFile = {"HRD":"HDS_new_pump.docx", "HRS":"HRS_new_pump.docx", "HTP":"HTP_new_pump.docx", "HTR":"HTR_new_pump.docx", \
           "PRS":"PRS_new_pump.docx", "RISK":"RiskAnalysis_Pump.docx", "SDS":"SDS_New_pump_x04.docx", \
           "ACE":"SRS_ACE_Pump_X01.docx", "BOLUS":"SRS_BolusCalc_Pump_X04.docx", "SRS":"SRS_DosingAlgorithm_X03.docx", \
           "SVAL":"SVaP_new_pump.docx", "SVATR":"SVaTR_new_pump.docx", "UT":"SVeTR_new_pump.docx", "URS":"URS_new_pump.docx"}

filePath = "C:/Users/steph/OneDrive/Desktop/Docs_Project/"
```

These are lists used in functions below to hold different tags and tag descriptions

```
docFileList = list(docFile.keys())          # This is a list of all main tags found in each document
leadingTags = []                            # List of Leading Tags
trailingTags = []                           # List of Trailing Tags
relationalLeadingTags = []                  # List of Leading Tags excluding "RISK" and "URS" tags
tagDescriptions = []                        # List of all trags with descriptions
uniqueParentTags = []                       # Unique Trailing Tags List

def GetText(filename):                      # Opens the document and places each paragraph into a list
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    fullText = [ele for ele in fullText if ele.strip()]    # Eliminates empty paragraphs or spaces
    return fullText                         # List of all document content
```

```python
import docx
import re

docFile = {"HRD":"HDS_new_pump.docx", "HRS":"HRS_new_pump.docx", "HTP":"HTP_new_pump.docx", "HTR":"HTR_new_pump.docx", \
           "PRS":"PRS_new_pump.docx", "RISK":"RiskAnalysis_Pump.docx", "SDS":"SDS_New_pump_x04.docx", \
           "ACE":"SRS_ACE_Pump_X01.docx", "BOLUS":"SRS_BolusCalc_Pump_X04.docx", "SRS":"SRS_DosingAlgorithm_X03.docx", \
           "SVAL":"SVaP_new_pump.docx", "SVATR":"SVaTR_new_pump.docx", "UT":"SVeTR_new_pump.docx", "URS":"URS_new_pump.docx"}

filePath = "C:/Users/steph/OneDrive/Desktop/Docs_Project/"

docFileList = list(docFile.keys())          # This is a list of all main tags found in each document
leadingTags = []                            # List of Leading Tags
trailingTags = []                           # List of Trailing Tags
relationalLeadingTags = []                  # List of Leading Tags excluding "RISK" and "URS" tags
tagDescriptions = []                        # List of all trags with descriptions
uniqueParentTags = []                       # Unique Trailing Tags List
```

Function that reads a document and **returns** a list containing each line of text

```python
def GetText(filename):                      # Opens the document and places each paragraph into a list
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    fullText = [ele for ele in fullText if ele.strip()]   # Eliminates empty paragraphs or spaces
    return fullText                         # List of all document content
```

```
import docx
import re

docFile = {"HRD":"HDS_new_pump.docx", "HRS":"HRS_new_pump.docx", "HTP":"HTP_new_pump.docx", "HTR":"HTR_new_pump.docx", \
           "PRS":"PRS_new_pump.docx", "RISK":"RiskAnalysis_Pump.docx", "SDS":"SDS_New_pump_x04.docx", \
           "ACE":"SRS_ACE_Pump_X01.docx", "BOLUS":"SRS_BolusCalc_Pump_X04.docx", "SRS":"SRS_DosingAlgorithm_X03.docx", \
           "SVAL":"SVaP_new_pump.docx", "SVATR":"SVaTR_new_pump.docx", "UT":"SVeTR_new_pump.docx", "URS":"URS_new_pump.docx"}

filePath = "C:/Users/steph/OneDrive/Desktop/Docs_Project/"

docFileList = list(docFile.keys())          # This is a list of all main tags found in each document
leadingTags = []                            # List of Leading Tags
trailingTags = []                           # List of Trailing Tags
relationalLeadingTags = []                  # List of Leading Tags excluding "RISK" and "URS" tags
tagDescriptions = []                        # List of all trags with descriptions
uniqueParentTags = []                       # Unique Trailing Tags List

def GetText(filename):                      # Opens the document and places each paragraph into a list
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    fullText = [ele for ele in fullText if ele.strip()]    # Eliminates empty paragraphs or spaces
    return fullText                         # List of all document content
```

List comprehension used to strip empty string data

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Iterates through list of all tags

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Uses tag as key to access corresponding document value from the docFile dictionary

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Iterates through the list returned by GetText( ) function containing line text

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

The SRS tag has 3 subtags (AID, BOLUS and ACE). The if statement runs only when BOLUS and ACE are found

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

These two blocks of code are identical except that one looks for SRS:BOLUS and SRS:ACE and the other looks for any tag

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Does a regular expression search for a pattern starting with characters followed by either a colon or space, followed by the SRS tag, and followed by a colon or space.

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Adds the index to the ind list for the tag used in the search pattern in the textList.

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Adds the leading tag with description corresponding trailing tags to the tagDescription list

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

The findall( ) function returns the pattern to be searched

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

The pattern should match a non-whitespace character, followed by either a space or a colon, followed by the tag name, followed by a space or a colon and ending in a non-whitespace character.

A non-whitespace character is any letters of symbols that is not a space. The search stops when it sees a space.

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Pulls the element out of the list by using y[0] and appends the leading tag found to the leadingTags list.

```
def GetLeadingTags():                          # Returns only valid parent tags
    for tag in docFileList:                    # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":                  # SRS has three additional tags - AID, BOLUS and ACE
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):   # This if block does the same as the else block
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + "SRS" + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    tagDescriptions.append(t)
                    y = re.findall('\S*[:\s]' + re.escape(tag) + '[:\s]\S*', t)
                    leadingTags.append(y[0])
                index = index + 1
            #print(ind)
        leadTagsAndDescriptions = [leadingTags, tagDescriptions]
    return leadTagsAndDescriptions
```

Creates a composite list (two lists in one) containing both the leading tags and corresponding tag descriptions and returns them.

```
def GetTrailingTags():                                  # Returns only valid child tags
    for tag in docFileList:                             # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        unique_tags = []
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):
                    ind.append(index)
                    y = re.findall('[\[{].+[\]}]', t)
                    if len(y) != 0:
                        trailingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    y = re.findall('[\[{].+[\]}]', t)
                    if len(y) != 0:
                        trailingTags.append(y[0])
                index = index + 1
    return trailingTags
```

This function goes through the exact same process as the GetLeadingTags document with slight changes.

```
def GetTrailingTags():                                  # Returns only valid child tags
    for tag in docFileList:                             # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        unique_tags = []
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):
                    ind.append(index)
                    y = re.findall('[\[{].+[\]}]', t)
                    if len(y) != 0:
                        trailingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    y = re.findall('[\[{].+[\]}]', t)
                    if len(y) != 0:
                        trailingTags.append(y[0])
                index = index + 1
    return trailingTags
```

The pattern search starts with either a bracket of curly brace, followed by any number of characters, and ending with either a bracket or curly brace.

```
def GetTrailingTags():                                  # Returns only valid child tags
    for tag in docFileList:                             # Tags are used to open the corresponding file
        textList = GetText(filePath + docFile[tag])
        unique_tags = []
        index = 0
        ind = []
        for t in textList:
            if tag == "BOLUS" or tag == "ACE":
                if re.search('.*[:\s]' + "SRS" + '[:\s]', t):
                    ind.append(index)
                    y = re.findall('[\[{].+[\]}]', t)
                    if len(y) != 0:
                        trailingTags.append(y[0])
                index = index + 1
            else:
                if re.search('.*[:\s]' + re.escape(tag) + '[:\s]', t):
                    ind.append(index)
                    y = re.findall('[\[{].+[\]}]', t)
                    if len(y) != 0:
                        trailingTags.append(y[0])
                index = index + 1
    return trailingTags
```

The trailing tags found get added to the trailingTags list and returned by the function

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags

def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

This function creates a Leading Tags list but excludes the RISK and URS tags because they do not have any parent tags associated with them.

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags

def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

This function calls the GetLeadingTags function and saves the list of Leading tags to the leadTagsList list.

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags

def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

This line searches the RISK and URS tags and excludes them from being saved into the relationalLeadingTags list.

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags

def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

This function creates a unique list of all the trailing tags found

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags


def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

This function calls the GetTailingTags( ) function and stores the trailing tags into the tralingTagsList

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags


def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

Iterates through all the trailing tags and deletes the opening and closing brackets.

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags

def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

This splits the tags into individual tags entries if there are multiple tags found

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags

def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

Saves each item into a separate list named uniqueParentTags

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags


def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

Uses the set( ) method to eliminate all duplicate tags.

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags


def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

Deletes NA, PASS and FAIL from the tags list

```
def GetRelLeadTags():
    leadTagsList = GetLeadingTags()
    for tag in leadTagsList[0]:
        if re.search('[:\s]' + "RISK" + '[:\s]', tag) or re.search('[:\s]' + "URS" + '[:\s]', tag):
            pass
        else:
            relationalLeadingTags.append(tag)
    return relationalLeadingTags

def GetUniqueParentTags():
    trailingTagsList = GetTrailingTags()
    for tags in trailingTagsList:
        tags = tags.replace('[', "").replace(']', "")
        tags = tags.split()
        for item in tags:
            uniqueParentTags.append(item)
    #uniqueTags = uniqueTags.sort()
    uniqueTags = list(set(uniqueParentTags))
    uniqueTags.remove('{NA}')
    uniqueTags.remove('{PASS}')
    uniqueTags.remove('{FAIL}')
    uniqueTags.sort()
    return uniqueTags
```

Sorts and returns the list in alphabetical order

```
def GetVerifiedUniqueTrailingTags():          # Returns only valid tags and extracts Orphan tags
    verifiedTrailingTags = []
    orphanTags = []
    tempTags = GetUniqueTrailingTags()
    for tt in tempTags:
        tag = re.findall('[:\s]' + ".*" + '[:\s]', tt)
        tag = tag[0].replace(':', "")
        if tag in docFileList:
            verifiedTrailingTags.append(tt)
        else:
            orphanTags.append(tt)
    verifiedTags = [verifiedTrailingTags,orphanTags]
    return verifiedTags
```

Calls sorted unique trailing tags list

```
def GetVerifiedUniqueTrailingTags():          # Returns only valid tags and extracts Orphan tags
    verifiedTrailingTags = []
    orphanTags = []
    tempTags = GetUniqueTrailingTags()
    for tt in tempTags:
        tag = re.findall('[:\s]' + ".*" + '[:\s]', tt)
        tag = tag[0].replace(':', "")
        if tag in docFileList:
            verifiedTrailingTags.append(tt)
        else:
            orphanTags.append(tt)
    verifiedTags = [verifiedTrailingTags,orphanTags]
    return verifiedTags
```

Gets tag string from list and deletes colons from each end.

```
def GetVerifiedUniqueTrailingTags():          # Returns only valid tags and extracts Orphan tags
    verifiedTrailingTags = []
    orphanTags = []
    tempTags = GetUniqueTrailingTags()
    for tt in tempTags:
        tag = re.findall('[:\s]' + ".*" + '[:\s]', tt)
        tag = tag[0].replace(':', "")
        if tag in docFileList:
            verifiedTrailingTags.append(tt)
        else:
            orphanTags.append(tt)
    verifiedTags = [verifiedTrailingTags,orphanTags]
    return verifiedTags
```

Checks each tag against a valid list named docFileList and adds the valid tags to a list otherwise appends the tags to and orphan tag list

Returns both the verified tags list and orphan list