RSS Feed Reader with Python

Although the author and publisher have made every effort to ensure that the information in this writing was correct at press time, the author and publisher do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause.

paid link

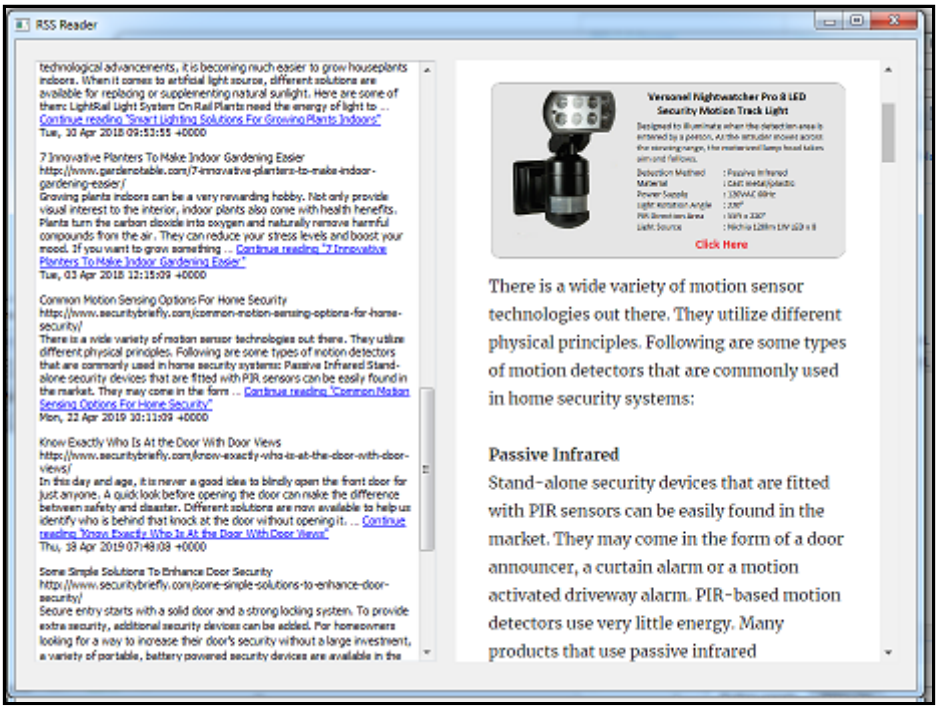
Many RSS Feed Readers are available out there. However, by creating your own, your RSS reader will be fully customizable. If you intend to build your own, this short writing can be a starting point.

Here is the program:

```
#from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5 import QtWidgets
from PyQt5 import QtCore #QCoreApplication, QUrl
from PyQt5 import QtWebEngineWidgets
from urllib.request import urlopen
from bs4 import BeautifulSoup
import sys

class App(QtWidgets.QMainWindow):
    def __init__(self,url_list):
        super().__init__()
        self.title = 'RSS Reader'
        self.left = 10
        self.top = 10
        self.width = 900
        self.height = 700
        self.url_list = url_list
        self.initUI()
    def initUI(self):
        #self.get_news('http://rss.cnn.com/rss/edition.rss')
        self.get_news()
```

```
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)

        self.txtView = QtWidgets.QTextBrowser(self)
        self.txtView.move(20,20)
        self.txtView.resize(400,600)
        self.txtView.setSource(QtCore.QUrl('temp.html'))
        self.txtView.setOpenLinks(False)
        self.txtView.anchorClicked.connect(self.on_anchor_clicked)
        self.txtView.show()

        self.webView2 = QtWebEngineWidgets.QWebEngineView(self)
        self.webView2.move(440,20)
        self.webView2.resize(440,600)
        self.webView2.loadProgress.connect(print)
        #self.webView2.load(QUrl(''))
        self.webView2.show()

        self.show()
    def on_anchor_clicked(self,url):
        url = str(url.toString())
        self.webView2.load(QtCore.QUrl(url))
    def get_news(self):
        item_list = []
        for url in self.url_list:
            xml_url = urlopen(url)
            xml_page = xml_url.read()
            xml_url.close()

            soup = BeautifulSoup(xml_page, 'xml')
            xml_content = soup.findAll('item')

            for item in xml_content:
                item_list.append('<p>')
                item_list.append('<div>'+item.title.text+'</div>')     #get the text inside tag
title
                item_list.append('<div><a href='+item.link.text+'>'+item.link.text+'</a></div>')
                if item.description:                                  #if description exists
                    item_list.append('<div>'+item.description.text+'</div>')
                if item.pubDate:
                    item_list.append('<div>'+item.pubDate.text+'</div>')
                item_list.append('</p>')

        with open('temp.html','w+') as file:
            file.write('<html> ')
            file.write('<body> ')
            for item in item_list:
                file.write(item+' ')
            file.write('</body> ')
            file.write('</html> ')
            file.close()


if __name__=='__main__':
    urls = [
            'http://www.gardenotable.com/feed/rss2/',
            'http://www.securitybriefly.com/feed/rss2/'
           ]
    app = QtCore.QCoreApplication.instance()
    if app is None:
        app = QtWidgets.QApplication(sys.argv)
    ex = App(urls)
    sys.exit(app.exec_())
```

Here is the screenshot:



Explanation:

```
if __name__=='__main__':
    urls = [
            'http://www.gardenotable.com/feed/rss2/',
            'http://www.securitybriefly.com/feed/rss2/'
           ]
    app = QtCore.QCoreApplication.instance()
    if app is None:
        app = QtWidgets.QApplication(sys.argv)
    ex = App(urls)
    sys.exit(app.exec_())
```

The variable *urls* is a list of rss feed urls. Change the values as you like. The variable is passed to the class *App*.

```
def __init__(self,url_list):
        super().__init__()
        self.title = 'RSS Reader'
        self.left = 10
        self.top = 10
        self.width = 900
        self.height = 700
        self.url_list = url_list
        self.initUI()
```

The *__init__* method set up initial values for the object, including window's size and position. It then calls *initUI()* to build the window and elements.

```
def initUI(self):
        self.get_news()

        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)

        self.txtView = QtWidgets.QTextBrowser(self)
        self.txtView.move(20,20)
        self.txtView.resize(400,600)
        self.txtView.setSource(QtCore.QUrl('temp.html'))
        self.txtView.setOpenLinks(False)
        self.txtView.anchorClicked.connect(self.on_anchor_clicked)
        self.txtView.show()

        self.webView2 = QtWebEngineWidgets.QWebEngineView(self)
```

```
        self.webView2.move(440,20)
        self.webView2.resize(440,600)
        self.webView2.loadProgress.connect(print)
        #self.webView2.load(QUrl(''))
        self.webView2.show()

        self.show()
```

In the *initUI()*, method *get_news()* is first called to get the content of the RSS feeds. The content is then displayed in *QTextBrowser*, a light browser for text and simple tags. It is worth noting that there are two classes in Qt that can display HTML content: *QTextBrowser* and *QWebEngineView*. Both are used in this program. When one of the links in the *QTextBrowser* is clicked, the output (the corresponding webpage) is displayed in *QWebEngineView*, a 'real' browser which is designed for complex layouts.

```
self.txtView.setSource(QtCore.QUrl('temp.html'))
```

When the above line of code is encountered by Python, RSS feed content, which is saved in a file named *temp.html*, is retrieved. The file *temp.html* itself is created when the method *get_news()* is called.

```
self.txtView.setOpenLinks(False)
```

What that line accomplishes is to prevent webpage to be loaded in the *QTextBrowser* when a link in clicked. We want the webpage to be displayed in the more heavy duty browser *QWebEngineView*.

```
self.txtView.anchorClicked.connect(self.on_anchor_clicked)
```

In the above line, *QTextBrowser* is 'connected' to *QWebEngineView*. Everytime a link in *QTextBrowser* is clicked, the method *on_anchor_clicked* is called, the corresponding webpage is loaded in the *QWebEngineView*.

```
self.webView2.loadProgress.connect(print)
```

Use this code if you want the progress of webpage loading is displayed in the Python Console.

```
item_list = []
        for url in self.url_list:
            xml_url = urlopen(url)
            xml_page = xml_url.read()
            xml_url.close()

            soup = BeautifulSoup(xml_page, 'xml')
            xml_content = soup.findAll('item')

            for item in xml_content:
                item_list.append('<p>')
                item_list.append('<div>'+item.title.text+'</div>')     #get the text inside tag
title
                item_list.append('<div><a href='+item.link.text+'>'+item.link.text+'</a></div>')
                if item.description:                                   #if description exists
                    item_list.append('<div>'+item.description.text+'</div>')
                if item.pubDate:
                    item_list.append('<div>'+item.pubDate.text+'</div>')
                item_list.append('</p>')
```

This block of code gets XML content from the RSS pages. The results are stored in the variable *item_list*. Only common tags are taken into account (*title*,*link*,*description* and *pubDate*). Note that an RSS feed may use different tag names. Also note that not all of the items in the feed have description.

```
with open('temp.html','w+') as file:
            file.write('<html> ')
            file.write('<body> ')
            for item in item_list:
                file.write(item+' ')
            file.write('</body> ')
            file.write('</html> ')
            file.close()
```

The values of the variable *item_list* are then stored in the *temp.html* file.