Extracted from:

# Portable Python Projects

## Run Your Home on a Raspberry Pi



The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

# Portable Python Projects

## Run Your Home on a Raspberry Pi

Mike Riley

*edited by Jacquelyn Carter*

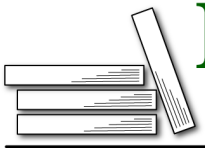
# Portable Python Projects

## Run Your Home on a Raspberry Pi

Mike Riley

The Pragmatic Bookshelf

Raleigh, North Carolina





For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Jacquelyn Carter
Copy Editor: L. Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.



ISBN-13: 978-1-68050-859-8
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—February 2022

The Pi 4 is remarkably powerful hardware, considering its diminutive size and relatively low cost. And yet, if you were to only use that computing power for monitoring occasional changes in sensors, a lot of compute cycles would be wasted. The purpose of this project is to give your Pi something to do during the off-hours that will hopefully make you more productive and even smarter when you're not sleeping.

The objective of the PiSpeak project is to retrieve the latest news using RSS feeds and use a text-to-speech (TTS) engine to convert those feeds into a single MP3 file that can be easily synced with a cloud storage service. Then that MP3 file can be downloaded to a smartphone and configured for playback in a smartphone MP3-player application. The script will also send an email notification to alert when new content is available along with a list of titles and links to those articles contained in the audio file for later reference.

This project features one of the longer scripts in the book. You can use whichever text editor you prefer, but I recommend using Microsoft Visual Code to help highlight the source code syntax, navigate, and debug your Python code. You can also use a Mac or Windows PC to write and test the script before ultimately deploying it to the Pi. Thanks to Python's portability, it's easy to code on one platform and run the script on a completely different OS and CPU architecture. Of course, if you prefer to work entirely on the Pi, that's fine too, since Visual Studio Code works identically on the Pi, albeit slower compared to more powerful Mac- or PC-hardware platforms. So put on your Python coder cap and let's get started.

## Setup

Here's what you need to build this project.

### Hardware

- Smartphone
- Speaker or headset to test audio playback
- (Optional) Mac or Windows PC for remote script development

### Software

- beautifulsoup4[1] Python library
- feedparser[2] Python library

1. https://pypi.org/project/beautifulsoup4/
2. https://pypi.org/project/feedparser/

- ffmpeg[3]
- mutagen[4] Python library
- Path (part of the standard Python library)
- Rclone (already installed in Chapter 2, Setting Up the Software, on page ?)
- shutil (part of the standard Python library)
- (Optional) DB Browser for SQLite[5]
- (Optional) SQLite Extension for VS Code[6]
- MP3 Audiobook Player Pro for iOS,[7] or
- Smart Book Player for Android[8]

Before we begin installing the list of software dependencies, let's specifically break down what we want to achieve.

1. Retrieve a list of recent articles posted to a selection of favorite news sources.
2. Aggregate these articles into a single body of text that can be processed for TTS.
3. TTS process the text and save the output as an MP3 file.
4. Accelerate the audio playback without altering the pitch, and convert the output to a new MP3 file.
5. Tag this new, sped-up MP3 file with title, artist, and cover art metadata for more meaningful rendering in audio playback applications.
6. Move this final MP3 file into a folder that synchronizes with a cloud storage service of choice, making it simple to retrieve the file from a smartphone or media playback system that may or may not be on our home or office network.
7. Send an email to notify you what new content is available for retrieval and the name of the timestamped MP3 file that contains it.
8. Download the MP3 file from your cloud storage provider via smartphone or computer.
9. Queue up, playback, and listen.

3. https://ffmpeg.org/
4. https://pypi.org/project/mutagen/
5. https://sqlitebrowser.org/
6. https://marketplace.visualstudio.com/items?itemName=alexcvzz.vscode-sqlite
7. https://apps.apple.com/us/app/mp3-audiobook-player-pro/id889580711
8. https://play.google.com/store/apps/details?id=ak.alizandro.smartaudiobookplayer

Implementing this sequential list of tasks is straightforward, especially since the apps and libraries to help us do this have already been built by dedicated members of the programming community. Let's begin with retrieving a news article listing using Really Simple Syndication (RSS).

## Feeding RSS

Back when RSS feeds were new, media companies were prominently promoting them as a means to subscribe and retrieve content updates. Over time, different means of notification and content consumption moved RSS into the background. There was also the potential confusion that nontechnical users had when clicking an RSS link only to see a bunch of XML code render in their browser window. Consequently, major media providers removed RSS references from their pages, but most news outlets still offer many of these RSS feeds if you know where to look for them.

For example, most blogging websites built on WordPress offer the ability to retreive their RSS feed by simply adding /feed/ after the site's domain name, such as the one for hackaday.com.[9] Depending on the web browser you're using, visiting that URL will render an XML document containing a number of various XML tags. Manually parsing and reading through such a document would be an eye-straining waste of time. Fortunately, Python can make these types of RSS document feeds intended for machines rather than humans to read much easier to consume and convert for our needs.

### Feeding the Details

While you could use Python's built-in library to retrieve and parse RSS feeds, a popular third-party library called feedparser already does that for us. Install it via the usual pip3 install command:

```
$ sudo pip3 install feedparser
```

You'll also need two other Python libraries. These will be used to remove any HTML tags, such as paragraph (<p>) or image (<image>) references. If we don't, these tags will be included in the text-to-speech conversion and make it unbearable to listen to (unless of course you enjoy hearing a series of angle brackets and tag names read back to you interspersed within the article text). A very popular tag parsing tool called Beautiful Soup 4, coupled with an easy-to-use Python HTML parser called lxml, will make the removal of these unwanted HTML tags as simple as one line of code. Install these libraries using pip3:

9. https://hackaday.com/feed/

```
$ sudo pip3 install beautifulsoup4

$ sudo pip3 install lxml
```

You'll find a vast sea of RSS feeds to choose from, but to focus on a demonstrative few, in this project we'll consume them from Pragmatic Bookshelf–sponsored developer conversation website Devtalk.com,[10] National Public Radio News,[11] and the Pi enthusiast website, Raspberry Projects.[12]

The following Python script will iterate through each of these feeds, retrieving and displaying the recently posted article titles, reference web page links, and descriptive content. Using your text editor of choice, create a file called feedtest.py and enter the following code:

pispeak/feedtest.py

```
1 from bs4 import BeautifulSoup
  import feedparser

2 URLS = ['https://forum.devtalk.com/latest.rss',
          'https://feeds.npr.org/1001/rss.xml',
          'https://projects-raspberry.com/news-updates/raspberry-pi-news/feed/',
          ]

3 for url in URLS:
      feed = feedparser.parse(url)

4     for entry in feed["entries"]:
          title = entry.get("title")
          link = entry.get("link")
5         description = BeautifulSoup(entry.get("description"), "lxml").text
6         print(title + '\n' + link + '\n' + description + '\n\n')
```

Let's quickly review the code used in this script before we attempt to run it.

1. Import the BeautifulSoup and feedparser Python libraries.
2. Store the three RSS feed URLs that will be parsed in a static URLS array.
3. Assign a single url from each URL stored in the URLS array, and process that RSS feed url by having feedparser parse the feed elements that can be later queried upon.
4. For each entry in the feed being parsed, extract the title, link and description elements.
5. Rather than just store the raw HTML description, remove the embedded HTML tags so they're not rendered when passing this string to the text-

10. https://forum.devtalk.com/latest.rss
11. https://feeds.npr.org/1001/rss.xml
12. https://projects-raspberry.com/news-updates/raspberry-pi-news/feed/

to-speech parser later in this project. Doing so will also make the output more human-readable as well.

❻ Combine the results of the title, link, and HTML-cleaned description into a string with appropriate line breaks, and print() out the results.

Save and run the feedtest.py file and execute it with the Python interpreter:

```
$ python3 feedtest.py
```

Assuming your code syntax is correct and your Pi is connected to the Internet, you should see a list of article titles and associated web links.

You now have a script that will quickly print out the latest newsfeed articles that have been posted to websites selected for this demonstration. That's pretty nifty. You also completed the project's first objective.

Next, you need to identify which of those listed articles were added since the last time the feedresults.py script was executed. Otherwise, you would have to listen to news items you previously heard sprinkled with any additions made to the feed since the last time the script was run. To accomplish that, retrieved key identifiers, namely the title and link entities, will be stored in a database. That database can be used to quickly check whether or not articles being retrieved have already been seen in previous feedresults.py results.