adif-edit.py

Description:

This script will modify ADIF (Amateur Data Interchange Format) files used for exchanging data between most amateur radio (ham radio) logging programs.

Use Case

I use a portable logging app on my phone and iPad to log contacts for POTA, SOTA, CHASING, CONTESTing. I also use contesting apps for various contests. Once the mission is complete, I export from the logging app to ADIF file format. In preparation for importing to my main logging program, “Amateur Contact Log” (ACLOG) developed by N4JFP. I use the ADIF file editor to quickly add a field with the activity name as the value. The field name is a user customizable field in ACLOG, called “OTHER” (Yes, at the time, I had no imagination).

The program can be used to add or edit ANY field in the ADIF file.

USAGE:

Assumes you initiate the python script using a python script engine on your computer.
eg: `“python adif-edit”`

```
adif-edit <input_file> [<output_file | -f>] <value> [<field_name> | -f]
```

If the python script is run without any arguments, it will prompt the user for each argument.

Because the application is mainly used to add a field to my files prior to import to populate the “other” field, it gives a warning if the user doesn’t enter an expected value, allowing the user to override and continue or enter a new value.

Prior to running, the user is presented with what the program will do, allowing the user to approve or quit the application.

- **input_file**
  The adif formatted file you want to process
  (if argument is blank, all arguments should be left blank causing the app to prompt the user for the input_file and other required arguments.)

- **output_file**
  File where you want the processed data to be placed
  (If the argument is blank all following arguments should be left blank, causing the app to prompt the user for the output file name and the rest of the arguments.)

  "-f" for full override mode, forcing default values,
  foo.adi for output_file, SOTA for activity, OTHER for field to be updated
  and it skips the confirmation step at the end.

- **value**
  The value that you want to be put in the field
  (If the argument left blank, all following arguments should be left blank, causing the app to prompt the user for the value and field_name)

- **field_name or "-f"**
  The name of the field that should be either edited (changed to the new value) or it's added if it's not there. If the user enters "-f" as the final argument, it will force the app to use the default field name of "OTHER".
  (If the argument is not passed, the user will be prompted for the value).

Examples

```
adif-edit foo.adi bar.adi sota other
```

- Takes the file foo.adi and either edits or adds a field called "OTHER" and populates that with the value "SOTA" and places it in a new file called bar.adi

```
adif-edit foo.adi bar.adi sota -f
```

- Takes the file foo.adi and either edits or adds a field called "OTHER" and populates that with "SOTA" placing the data in a new file called bar.adi.

  The "-f" in place of the field name forces the app to use the default field.

```
adif-edit foo.adi
```

- Uses foo.adi as the input file and prompts the user for the rest of the required arguments

Author:

Created by Chris Claborne, initially using claude.ai and then modified to fit my needs as well as fix a bug in the original AI developed code.

Go to HamNinja.com/logging to see how I generally manage logs.