How to process command line arguments?

Have you ever wondered how to process command line arguments in Python? Yeah, there's a module for that. It's called argparse, which is a replacement for optparse. In this article, we'll be taking a whirlwind tour of this helpful module. Let's start with something simple!

Getting Started

I have always found the simplest way to explain a coding concept is to show some code. So that's what we're going to do. Here's a super simple example that doesn't do much of anything:

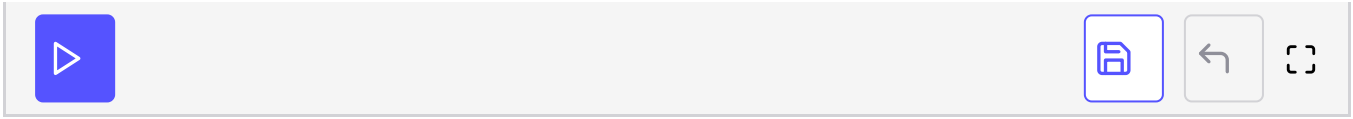
```
import argparse
parser = argparse.ArgumentParser(
         description="A simple argument parser",
         epilog="This is where you might put example usage"
     )

parser.print_help()
#usage:  __ed_file.py [-h]

#A simple argument parser

#optional arguments:
#  -h, --help  show this help message and exit

#This is where you might put example usage
```



Here we just import argparse and give it a description and set up a usage section. The idea here is that when you ask the program you are creating for help, it will tell you how to use it. In this case, it prints out a simple description, the default optional arguments ("-h" in this case) and example usage.

Now let's make this example a bit more concrete. You won't normally be

parsing arguments from the command-line after all. So we'll move the code

into a Python function inside of a Python file:

```
# arg_demo.py

import argparse


def get_args():
    """"""
    parser = argparse.ArgumentParser(
        description="A simple argument parser",
        epilog="This is where you might put example usage"
    )
    return parser.parse_args()

if __name__ == '__main__':
    get_args()
```

Now let's call the script from the command line:

```
python arg_demo.py -h
```

This will print out the help text like we saw earlier. Now let's learn about how to add some of our own custom arguments.