EPAM Python Software Engineer Training

Lesson 2: Argument Parsing

Contents

1 Course **1**

1.1 An argparse library 1

2 Tasks **1**

2.1 File Copy 1

2.2 Regexp 2

2.3 Phone Book 2

2.4 Phone Book Script 2

1 Course

1.1 An argparse library

In Python a typical library for command line argument parsing is *argparse*. It is a must for a Python program accepting command line arguments to parse them using an *argparse* library.

2 Tasks

2.1 File Copy

Implement a parametrized file copy operation accepting the following arguments:

- *input file path*: specified as either first argument, or through one of `-i, --input` options, required;
- *output file path*: specified as either second argument, or through one of `-o, --output` options, required;
- *buffer size in bytes*: specified as one of `-s, --buffer-size` options, optional, defaults to `8192`;
- *program help*: specified as one of `-h, --help` option; an application shall print a program help and exit if this option is specified;
- *verbose mode*: specified as one of `-v, --verbose` options, optional, defaults to `False`; an application shall print a dot per each file chunk copy in this mode;

In case of any argument error an application shall print an appropriate error and a short usage information (not help). Any other error shall be handled gracefully. An application shall return an appropriate exit code.

E.g. all of the below commands shall behave the same:

```
-ooutput.txt -iinput.txt

-o output.txt -s 8192 input.txt

input.txt output.txt
```

```
--input input.txt output.txt --buffer-size=8192
```

2.2 Regexp

Write an application that performs a set of operations on a given file defined in a `05-regexp` task. A file shall be specified as the first argument. All operations shall be executed in the same order as they appear in a command line. For example, `-a -o -a` means that a command defined for option `-a` shall be performed, followed by a command for option `-o` and a command for `-a` once again.

2.3 Phone Book

Write a simple phone book - an application storing a phone contacts information in a plain (readable) file format. There shall be a possibility to store several phone numbers for one person. It shall be possible to specify a file path to a phone book in command line arguments; by default it shall point to a `phonebook.txt` file in the same folder as an application.

An application shall support the following operations:

- add a new contact (a First and a Last names) with an arbitrary number of associated phone numbers;
- edit an existing contact information (modify a First or a Last name; add, delete, or modify an associated phone number);
- delete an existing contact information;
- print a list of first N contacts (unordered or sorted alphabetically);
- print a specific contact information given a First and a Last names;
- search a contact information (a First and a Last name) given a phone number (efficiency doesn't matter) and print a list of found contacts;

2.4 Phone Book Script

Not to be done. For an above application think of how would you implement a simple scripting language allowing to run several commands consequently within one application. Present a prototype doing this for one command (it would require too much time producing a full working solution).

> ***Hint***
>
> This can be achieved by a combination of *nargs* and *REMAINDER*...