CLI subcommands with precise help output

Python 2.x 2.7 , Python 3.x 3.4–3.5

Different ways to create subcommands like in hg or svn with the exact command line interface and help output as shown in Remarks section.

Parsing Command Line arguments covers broader topic of arguments parsing.

Examples

argparse (custom help formatter)

Extended version of https://stackoverflow.com/documentation/python/7701/cli-subcommands/25282/argparse-default-help-formatter that fixed help output.

```
import argparse
import sys

class CustomHelpFormatter(argparse.HelpFormatter):
    def _format_action(self, action):
        if type(action) == argparse._SubParsersAction:
            # inject new class variable for subcommand formatting
            subactions = action._get_subactions()
            invocations = [self._format_action_invocation(a) for a in subactions]
            self._subcommand_max_length = max(len(i) for i in invocations)

        if type(action) == argparse._SubParsersAction._ChoicesPseudoAction:
            # format subcommand help line
            subcommand = self._format_action_invocation(action) # type: str
            width = self._subcommand_max_length
            help_text = ""
            if action.help:
                help_text = self._expand_help(action)
            return "  {:{width}} -  {}\n".format(subcommand, help_text, width=width)

        elif type(action) == argparse._SubParsersAction:
            # process subcommand help section
            msg = '\n'
            for subaction in action._get_subactions():
                msg += self._format_action(subaction)
            return msg
        else:
            return super(CustomHelpFormatter, self)._format_action(action)


def check():
    print("status")
    return 0

parser = argparse.ArgumentParser(usage="sub <command>", add_help=False,
             formatter_class=CustomHelpFormatter)
```

Output without arguments:

```
usage: sub <command>

commands:

  status -  show status
  list   -  print list
```

argparse (default help formatter)

```
import argparse
import sys

def check():
    print("status")
    return 0

parser = argparse.ArgumentParser(prog="sub", add_help=False)
subparser = parser.add_subparsers(dest="cmd")

subparser.add_parser('status', help='show status')
subparser.add_parser('list', help='print list')

# hack to show help when no arguments supplied
if len(sys.argv) == 1:
    parser.print_help()
    sys.exit(0)
```

```
args = parser.parse_args()

if args.cmd == 'list':
    print('list')
elif args.cmd == 'status':
    sys.exit(check())
```

Output without arguments:

```
usage: sub {status,list} ...

positional arguments:
  {status,list}
    status       show status
    list         print list
```

Pros:

- comes with Python
- option parsing is included

Native way (no libraries)

```
"""
usage: sub <command>

commands:

  status -  show status
  list   -  print list
"""

import sys

def check():
    print("status")
    return 0

if sys.argv[1:] == ['status']:
    sys.exit(check())
elif sys.argv[1:] == ['list']:
    print("list")
else:
    print(__doc__.strip())
```

Output without arguments:

```
usage: sub <command>

commands:

  status -  show status
  list   -  print list
```

Pros:

- no deps
- everybody should be able to read that
- complete control over help formatting

Syntax

Parameters

Remarks

Different ways to create subcommands like in `hg` or `svn` with the command line interface shown in the help message:

```
usage: sub <command>

commands:

  status -  show status
  list   -  print list
```