Build a Command Line Interface

Objectives:

At the end of this episode, I will be able to:

1. Discuss the features of a command-line interface.
2. Build a simple command-line interface using argparse.

Snippets

episode cli.py

```
    #!/usr/bin/env python

import argparse
import commands

def main():
    """Main entrypoint of the cli."""
    parser = argparse.ArgumentParser()
    subparsers = parser.add_subparsers(dest='command')

    # first command
    add = subparsers.add_parser(commands.ADD)
    add.add_argument("numbers", nargs="+", type=int)

    #second command
    sub = subparsers.add_parser(commands.SUBTRACT)
    sub.add_argument('numbers', nargs='+', type=int)

    args = parser.parse_args()
    if args.command == commands.ADD:
        result = sum(args.numbers)
        print(f"The sum of {args.numbers} is {result}.")
    elif args.command == commands.SUBTRACT:
        first, *rest = args.numbers
        result = first - sum(rest)
        print(f"The difference of {args.numbers} is {result}.")
    else:
        parser.print_help()


if __name__ == "__main__":
    main()
```

episode commands.py

```
    ADD = 'add'
SUBTRACT = 'subtract'
```

complete cli.py

```
    #!/usr/bin/env python
import commands
import argparse

def multiply(ns):
    """Multiplies all of the numbers contained in ns."""
```

```
    result = 1
    for n in ns:
        result *= n
    return result


def main():
    parser = argparse.ArgumentParser()
    subparsers = parser.add_subparsers(dest="command")

    add = subparsers.add_parser(commands.ADD)
    add.add_argument(
        "numbers",
        nargs="+",
        type=int,
    )

    sub = subparsers.add_parser(commands.SUBTRACT)
    sub.add_argument(
        "numbers",
        nargs="+",
        type=int,
    )

    mul = subparsers.add_parser(commands.MULTIPLY)
    mul.add_argument("numbers", nargs="+", type=int)

    div = subparsers.add_parser(commands.DIVIDE)
    div.add_argument("numbers", nargs="+", type=int)


    args = parser.parse_args()

    if args.command == commands.ADD:
        result = sum(args.numbers)
        operation = " + ".join(str(i) for i in args.numbers)
        print(f"The result of {operation} is {result}.")
    elif args.command == commands.SUBTRACT:
        first, *rest = args.numbers
        result = first - sum(rest)
        operation = " - ".join(str(i) for i in args.numbers)
        print(f"The result of {operation} is {result}.")
    elif args.command == commands.MULTIPLY:
        result = multiply(args.numbers)
        operation = " x ".join(str(i) for i in args.numbers)
        print(f"The result of {operation} is {result}.")
    elif args.command == commands.DIVIDE:
        first, *rest = args.numbers
        result = first / multiply(rest)
        operation = " ÷ ".join(str(i) for i in args.numbers)
        print(f"The result of {operation} is {result}.")
    else:
        parser.print_help()


if __name__ == "__main__":
    main()
```

complete commands.py

```
    ADD = "add"
SUBTRACT = "sub"
MULTIPLY = "mul"
DIVIDE = "div"
```

External Resources:

During this episode, you can reference the following external resources for supplementary tools and information:

- (S) Python argparse Module