**Building a CLI to Reverse Files**

Narzędzie, które zbudujemy w tym filmie, będzie musiało wykonać następujące czynności:

1. Wymagaj argumentu nazwy pliku, aby wiedział, który plik ma zostać odczytany.

2. Wydrukuj zawartość pliku wstecz (najpierw dół skryptu, każda linia wydrukowana wstecz)

3. Podaj tekst pomocy i dokumentację, gdy otrzyma flagę --help.

4. Zaakceptuj opcjonalną flagę --limit lub -l, aby określić liczbę linii do odczytania z pliku.

5. Zaakceptuj flagę --version, aby wydrukować bieżącą wersję narzędzia.

Brzmi to trochę, ale na szczęście moduł argparse sprawi, że większość tego trywialnego. Będziemy stopniowo tworzyć ten skrypt, gdy dowiemy się, co może zrobić argparse.ArgumentParser. Zacznijmy od zbudowania ArgumentParser z naszym wymaganym argumentem:

#!/usr/bin/env python3.6

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument('filename', help='the file to read')

args = parser.parse\_args()

print(args)

Tutaj stworzyliśmy instancję ArgumentParser bez żadnych argumentów. Następnie użyjemy metody add\_argument w celu określenia argumentu pozycyjnego o nazwie filename i udostępnienia tekstu pomocy za pomocą argumentu pomocy. Na koniec mówimy parserowi, aby przeanalizował argumenty ze stdin za pomocą metody parse\_args i zapisał z przeanalizowanych argumentów jako zmienną args.

Uczyńmy nasz skrypt wykonywalnym i wypróbujmy go bez żadnych argumentów:

$ chmod u+x ~/bin/reverse-file

$ reverse-file

usage: reverse-file [-h] filename

reverse-file: error: the following arguments are required: filename

Ponieważ nazwa pliku jest wymagana i nie podano obiektu ArgumentParser, rozpoznał problem i zwrócił użyteczny komunikat o błędzie. To cudownie! Możemy również zobaczyć, że wygląda na to, że ma już flagę -h, spróbujmy teraz:

$ reverse-file -h

usage: reverse-file [-h] filename

positional arguments:

filename the file to read

optional arguments:

-h, --help show this help message and exit

Wygląda na to, że już zajęliśmy się naszym wymogiem dostarczenia tekstu pomocy. Ostatnią rzeczą, którą musimy przetestować, jest to, co się dzieje, gdy podajemy parametr nazwy pliku:

$ reverse-file testing.txt

Namespace(filename='testing.txt')

Widzimy tutaj, że argumenty w naszym skrypcie są obiektem przestrzeni nazw. Jest to prosty typ obiektu, którego jedynym celem jest trzymanie nazwanych informacji z naszego ArgumentParser jako atrybutów. Jedynym atrybutem, o który go poprosiliśmy, jest atrybut nazwa\_pliku, i widzimy, że ustawił on wartość na 'testing.txt', ponieważ to właśnie przekazaliśmy. Aby uzyskać dostęp do tych wartości w naszym kodzie, będziemy łączyć się w łańcuch wyłącz nasz obiekt argumentów z kropką:

>>> args.filename

'testing.txt'

**Adding Optional parameters**

Zajmowaliśmy się już dwoma z pięciu wymagań, które ustawiliśmy dla tego skryptu; kontynuujmy dodając opcjonalne flagi do naszego parsera, a następnie zakończymy implementując prawdziwą logikę skryptu. Musimy dodać flagę --limit z aliasem -l.

#!/usr/bin/env python3.6

import argparse

parser = argparse.ArgumentParser(description='Read a file in reverse')

parser.add\_argument('filename', help='the file to read')

parser.add\_argument('--limit', '-l', type=int, help='the number of lines to read')

args = parser.parse\_args()

print(args)

Aby określić, że argument jest flagą, musimy umieścić dwa myślniki na początku nazwy flagi. Użyliśmy opcji type dla add\_argument, aby stwierdzić, że chcemy, aby wartość została przekonwertowana na liczbę całkowitą, i określiliśmy krótszą wersję flagi jako drugi argument.

Oto, jak wygląda teraz argument:

$ reverse-file --limit 5 testing.txt

Namespace(filename='testing.txt', limit=5)

Następnie dodamy flagę --version. Ten będzie trochę inny, ponieważ użyjemy opcji akcji, aby określić ciąg do wydrukowania po otrzymaniu tej flagi:

#!/usr/bin/env python3.6

import argparse

parser = argparse.ArgumentParser(description='Read a file in reverse')

parser.add\_argument('filename', help='the file to read')

parser.add\_argument('--limit', '-l', type=int, help='the number of lines to read')

parser.add\_argument('--version', '-v', action='version', version='%(prog)s 1.0')

args = parser.parse\_args()

print(args)

Używa wbudowanej wersji działania, którą znaleźliśmy w dokumentacji.

Oto, co otrzymujemy, gdy testujemy flagę --version:

$ reverse-file --version

reverse-file 1.0

*Uwaga: Zwróć uwagę, że wykonała akcję wersji i nie kontynuowała przechodzenia przez skrypt.*

#### Adding Our Business Logic

W końcu mamy szansę skorzystać z naszej wiedzy o pliku IO w skrypcie:

#!/usr/bin/env python3.6

import argparse

parser = argparse.ArgumentParser(description='Read a file in reverse')

parser.add\_argument('filename', help='the file to read')

parser.add\_argument('--limit', '-l', type=int, help='the number of lines to read')

parser.add\_argument('--version', '-v', action='version', version='%(prog)s 1.0')

args = parser.parse\_args()

with open(args.filename) as f:

lines = f.readlines()

lines.reverse()

if args.limit:

lines = lines[:args.limit]

for line in lines:

print(line.strip()[::-1])

Oto, co otrzymujemy, kiedy przetestujemy to na pliku xmen\_base.txt

$ reverse-file xmen\_base.txt

gnihtemoS

reivaX rosseforP

relwarcthgiN

pohsiB

spolcyC

enirevloW

mrotS

~ $ reverse-file -l 2 xmen\_base.txt

gnihtemoS

reivaX rosseforP