**Handling Errors with try/except/else/finally**

Co dzieje się w naszym skrypcie odwrotnego pliku, jeśli nazwa pliku nie istnieje? Dajmy mu szansę:

$ reverse-file fake.txt

Traceback (most recent call last):

File "/home/user/bin/reverse-file", line 11, in

with open(args.filename) as f:

FileNotFoundError: [Errno 2] No such file or directory: 'fake.txt'

Ten FileNotFoundError to coś, czego możemy się spodziewać dość często, a nasz skrypt powinien poradzić sobie z tą sytuacją. Nasz parser tego nie złapie, ponieważ technicznie korzystamy z CLI poprawnie, więc musimy sobie z tym poradzić. Aby poradzić sobie z tymi błędami, użyjemy słów kluczowych try, z wyjątkiem i innych.

#!/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 verison 1.0')

args = parser.parse\_args()

try:

f = open(args.filename)

limit = args.limit

except FileNotFoundError as err:

print(f"Error: {err}")

else:

with f:

lines = f.readlines()

lines.reverse()

if limit:

lines = lines[:limit]

for line in lines:

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

Używamy instrukcji try do oznaczenia, że całkiem możliwe jest wystąpienie błędu w jej obrębie. Stamtąd możemy obsługiwać określone typy błędów za pomocą exceptkeyword (możemy mieć więcej niż jedno). W przypadku, gdy nie ma błędu, chcemy wykonać kod znajdujący się w bloku else. Jeśli chcemy wykonać jakiś kod, niezależnie od tego, czy wystąpił błąd, czy nie, możemy umieścić go w bloku na końcu naszego t, z wyjątkiem przepływu pracy.

Teraz, gdy próbujemy naszego skryptu z fałszywym plikiem, otrzymujemy znacznie lepszą odpowiedź:

$ reverse-file fake.txt

Error: [Errno 2] No such file or directory: 'fake.txt'