```
# course: cmps3500
# Activity 7
# date: 12/8/24
# username: mchitorog
# name: Mihail Chitorog
# file_name: PythonException.pdf
# description: Tracing python exceptions report
```

Explanation of Character Series Output in filewords.py Program

Program Purpose

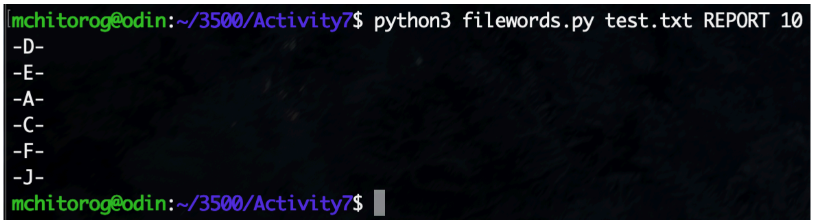
This program is designed to create a text file and fill it with multiple copies of a word. For example, we could use it to create a file containing the word "hello" written 3 times.

Here are the possible output sequences:

Successful Run

When everything works perfectly, we will see:

-D-
-E-
-A-
-C-
-F-
-J-



```
mchitorog@odin:~/3500/Activity7$ python3 filewords.py test.txt REPORT 10
-D-
-E-
-A-
-C-
-F-
-J-
mchitorog@odin:~/3500/Activity7$
```

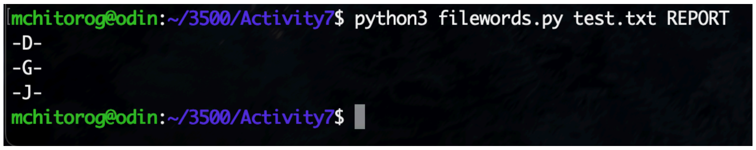
This happens when:

- We provide all three required arguments correctly
- The file doesn't already exist
- We have permission to create files in that location

Missing Arguments

If we don't provide enough arguments, we will see:

-D-
-G-
-J-



```
mchitorog@odin:~/3500/Activity7$ python3 filewords.py test.txt REPORT
-D-
-G-
-J-
mchitorog@odin:~/3500/Activity7$
```

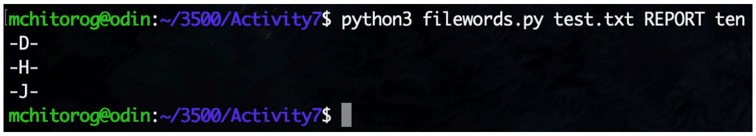
This happens when:

- We run the program without all three required arguments
- Example: “python3 filewords.py” or “python3 filewords.py test.txt REPORT”

Invalid Count Number

If we provide a non-number for the count, we will see:

-D-
-H-
-J-



```
mchitorog@odin:~/3500/Activity7$ python3 filewords.py test.txt REPORT ten
-D-
-H-
-J-
mchitorog@odin:~/3500/Activity7$ 
```

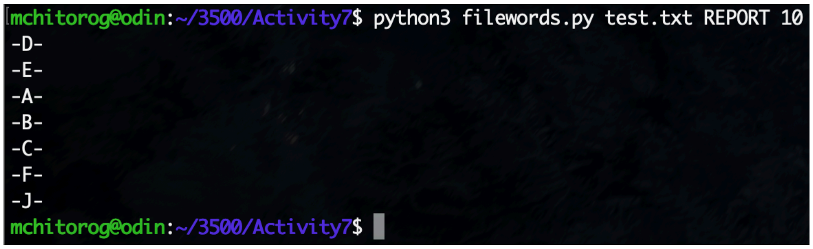
This happens when:

- The third argument isn't a valid number
- Example: "python3 filewords.py test.txt REPORT ten"

File Already Exists (Version 1)

If the file already exists, we might see:

-D-
-E-
-A-
-B-
-C-
-F-
-J-



```
mchitorog@odin:~/3500/Activity7$ python3 filewords.py test.txt REPORT 10
-D-
-E-
-A-
-B-
-C-
-F-
-J-
mchitorog@odin:~/3500/Activity7$ 
```

This happens when:

- The target file already exists
- The error is caught in the print_words function

File Already Exists (Version 2)

Alternatively, we might see:

-D-
-E-
-I-
-J-

This happens when:

- The target file already exists
- The error is caught in the main function

Catastrophic Error

If something completely unexpected goes wrong, you'll see:

-K-

This happens when:

- There's an error that isn't specifically caught by our other error handlers
- For example, if there's a permission error when trying to create the file

Why Different Sequences Happen

The program uses letters to mark different points in its execution:

- -D- marks the start of processing
- -E- marks successful argument processing
- -A- marks the start of file creation
- -B- marks a file already existing (caught in print_words)
- -C- marks completion of print_words attempt
- -F- marks successful completion of main function
- -G- marks missing argument error
- -H- marks invalid number error
- -I- marks file exists error (caught in main)
- -J- marks end of normal execution
- -K- marks catastrophic error

The sequence we see depends on where the program encounters problems. It's like a choose-your-own-adventure story - different problems lead to different paths through the code, resulting in different sequences of letters being printed.