# Question 3

Contents

[Question 3 1](#_Toc196328567)

[**Grammer** 1](#_Toc196328568)

[**Python Code** 2](#_Toc196328569)

[***File Collection Changes*** 3](#_Toc196328570)

[***Selector Changes*** 3](#_Toc196328571)

[***Main and FspowVisitor Changes*** 5](#_Toc196328572)

To start this question, I copied question 2 and made a new folder called Q3. I removed all the .class files and ran the setup commands again to generate fresh tokens, antlr python3 files, and class files.

I noted down what was required to be modified from the question 2 codebase. The codebase had to be extended to have the ability to provide the top 10 of a file collection. It had to filter the list by biggest, smallest, oldest, or newest. A test case file had to be created to test these features to ensure that the language worked as intended.

## **Grammer**

To start, I added a new filter to the selfilter list:



I added “top” as a filter. I allowed the filter to select a specific number (NUMBER) of files based on the attribute (topAttr). Where the NUMBER specifies how many files to select, and topAttr defines the attribute to use for the selection.

The addition of topAttr defines the attributes that can be used with the top filter. These attributes specify how the files should be ranked when selecting the top NUMBER of files. The options include:

1. Biggest: Selects the largest files by size.
2. Smallest: Selects the smallest files by size.
3. Oldest: Selects the oldest files by modification date.
4. Newest: Selects the newest files by modification date.

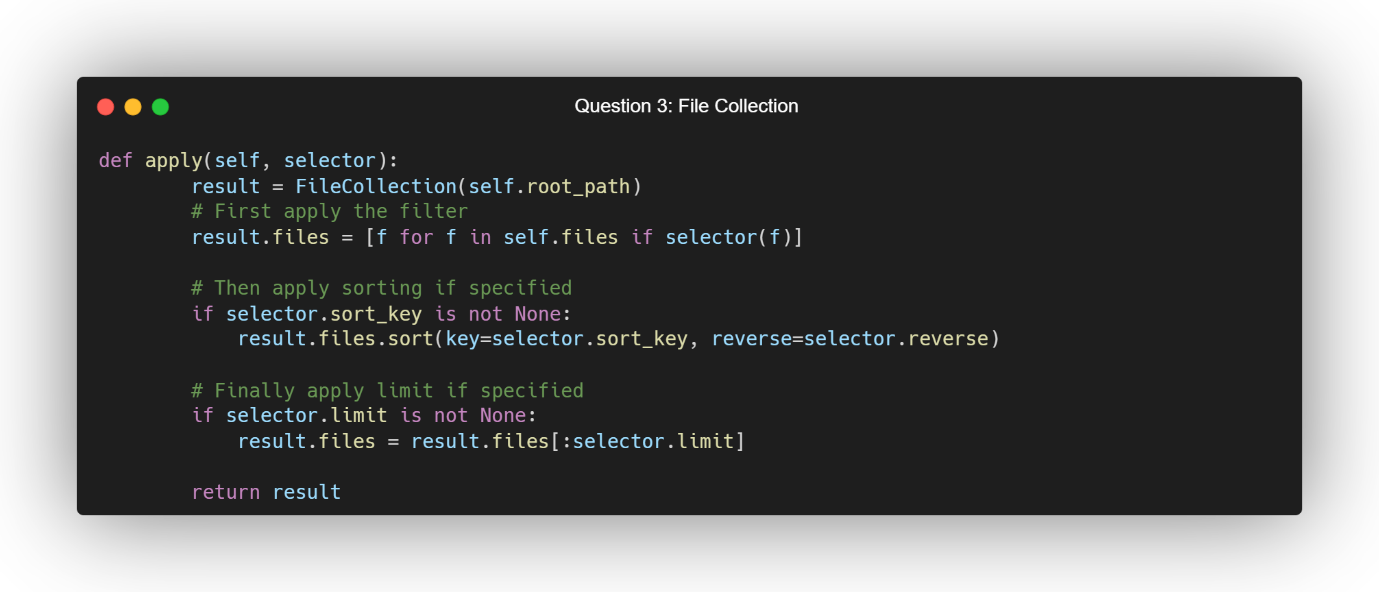
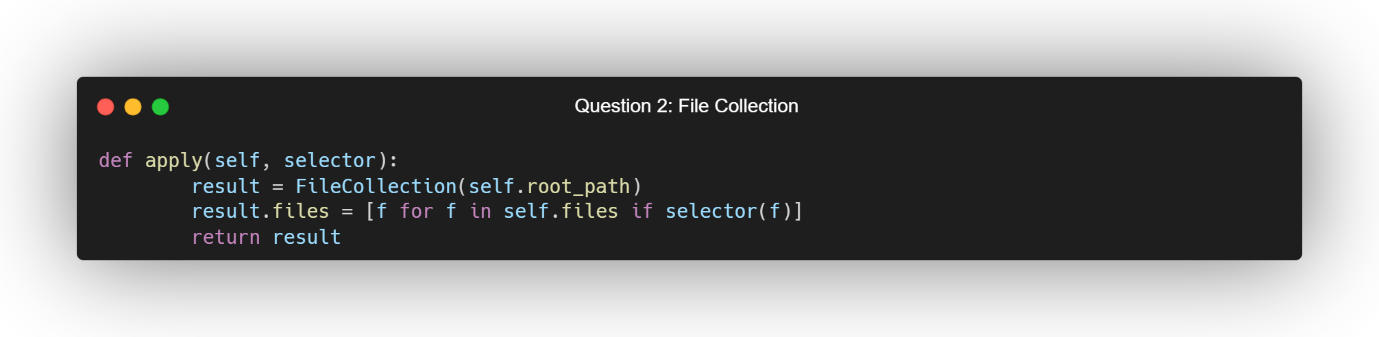
Some other changes I made to the fspow.g4 were adding a new Lexer rule for how commas are changed as I had some issues when running the grun command during testing.

COMMA: ',';

## **Python Code**

When beginning this question, I chose to only add the changes required and kept my deletions to a minimum as it could affect how the grammar operates. As I was reusing the files from question 2, I did not need to worry about editing the previous logic to account for the new use case. This reduced the time needed to read through the files and understand them as I had already made and tested the quality of them in the previous question.

### ***File Collection Changes***

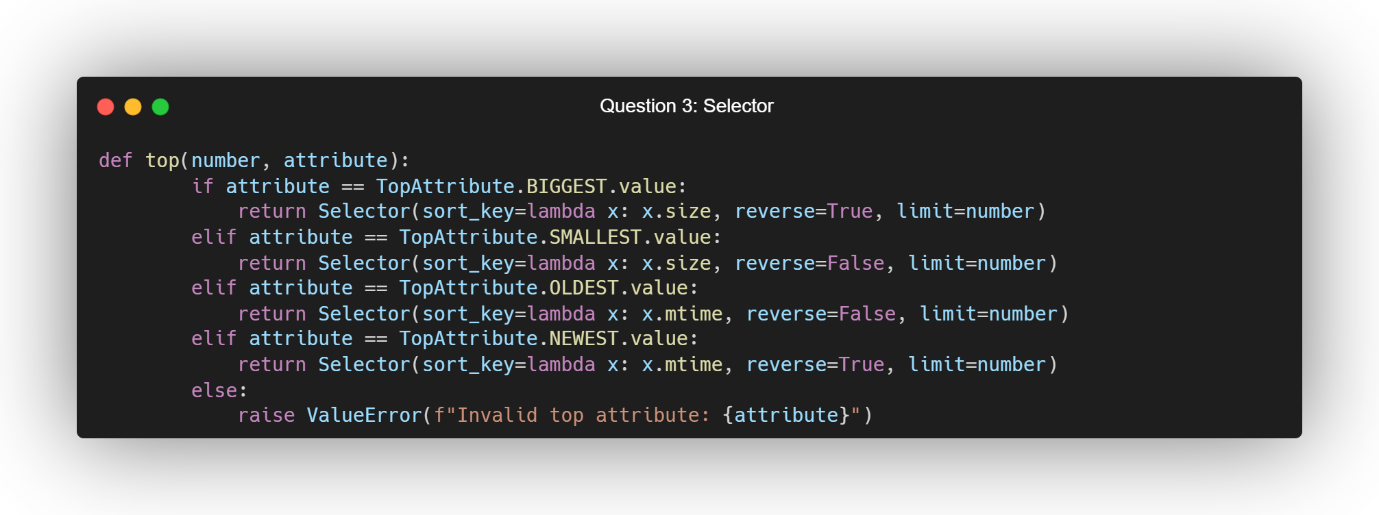
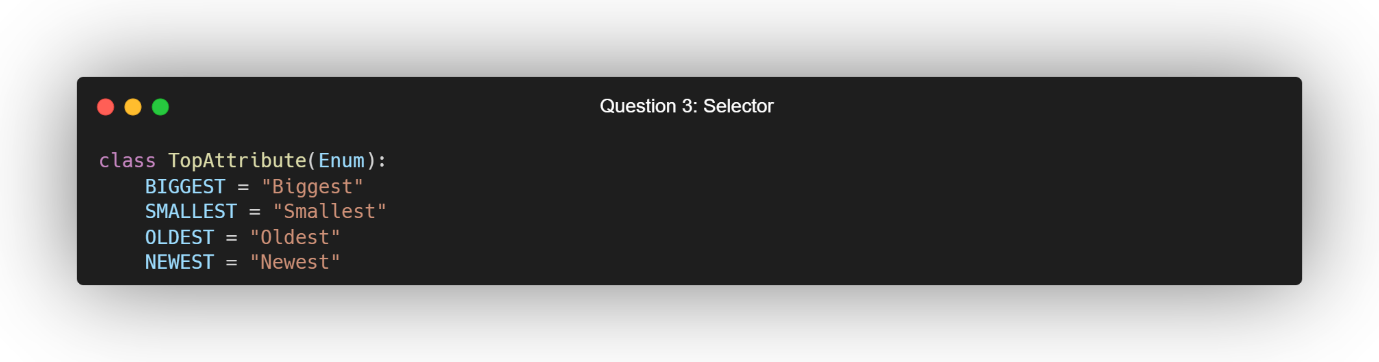
The file collection file stayed the same pretty much besides a modification to the apply function. The key difference of the update was added support for sorting and limiting, along with more complexity and a more advanced use case for operations like the top filter.

Since the use case for question 2 was simpler, the apply function did not need to be advanced. While on the other hand, because question 3 extends the functionality to include sorting and limiting, enabling a more advanced operation like the top filter were required in order to align with the questions new extended use case.

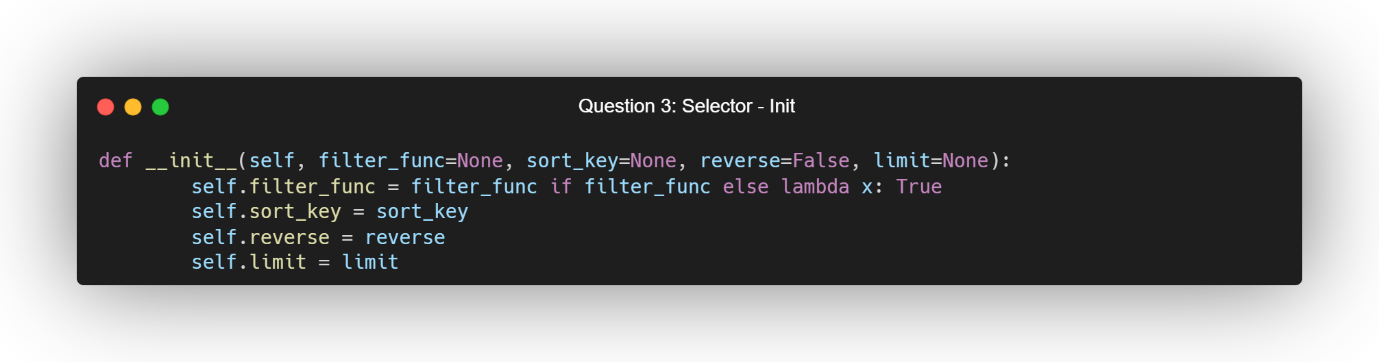
### ***Selector Changes***

The Selector class in question 3 includes new logic to handle the new use case functionality.

#### Top Method

A top method was added that allows selecting a specified number of files based on the attributes mentioned in fspow.g4. It uses the class called TopAttribute enum which defines the attributes for ranking and limiting results.

#### Sorting and Limiting

The constructor was updated to include a way to sort and limit. This was done by simply adding things like sort key, reverse and limit to the \_\_init\_\_ function. 

This allowed me to easily apply how to sort through files, if I wanted to reverse the order of the files, or limit how many files were being shown in the results. Compared to question 2, the additions of these parameters allow for a more advanced use case focusing on filtering.

Some other differences in the selector file include the difference focus on Filtering vs Ranking. In question 2, the focus was solely on filtering files based on conditions like name, size, and date. While in question 3, the focus is extended by adding ranking, sorting, and limiting capabilities to make it more versatile.

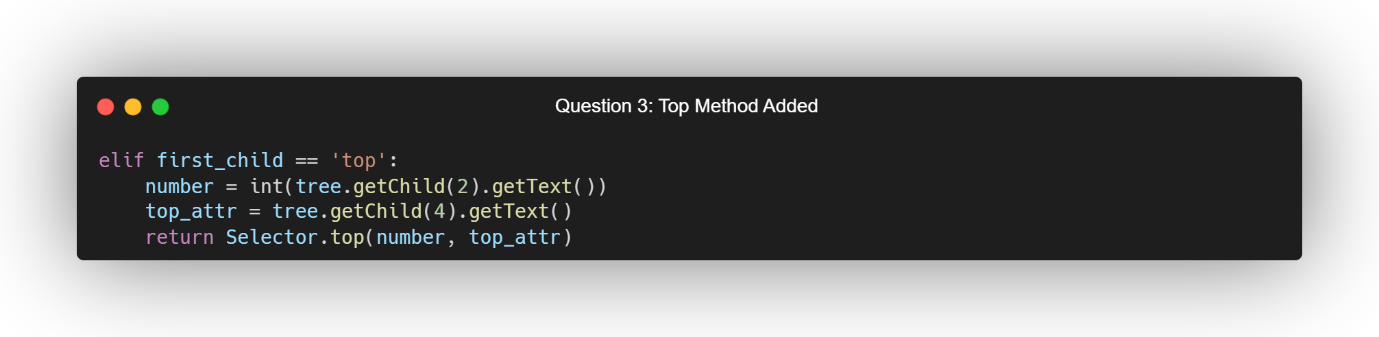
Overall, the structure of the code has been adjusted as well, the core filtering methods are identical in both versions, but addition of the top method and modification to the constructor to support ranking and limitations makes the file longer and more to maintain. The file attempts to stay optimized to improve the maintainability of the file, ensuring that features are understood by leaving comments on key sections of the code so if I were to look back on this code or a third party were to review the code, they would be able to understand what that section does without having to look at documentation or assume through the name of the function alone.

### ***Main and FspowVisitor Changes***

The main file is largely similar to the one in question 2, holding similar structure and functionality, but there are key differences related to the additional features to the FspowVistor class.

#### FspowVisitor

The FspowVisitor class was modified to support the top selector in the selfilter rule set. This was quite simple to include as it was similar to the surround logic.



This allows the user to create a Selector that selects the top files based on attributes mentioned in the selector. This addition adds support for the top selector, enabling ranking and limiting of results based on attributes given from the parameter.

This change is the only functional difference between question 2 and question 3 files.