# Brief Description

The primary goal of this code is to identify inverted repeats in DNA sequences, focusing on two types:

- Inverted repeats without spacers: where the two parts of the repeat are directly adjacent.

- Inverted repeats with spacers: where the two parts of the repeat have a certain number of bases between them.

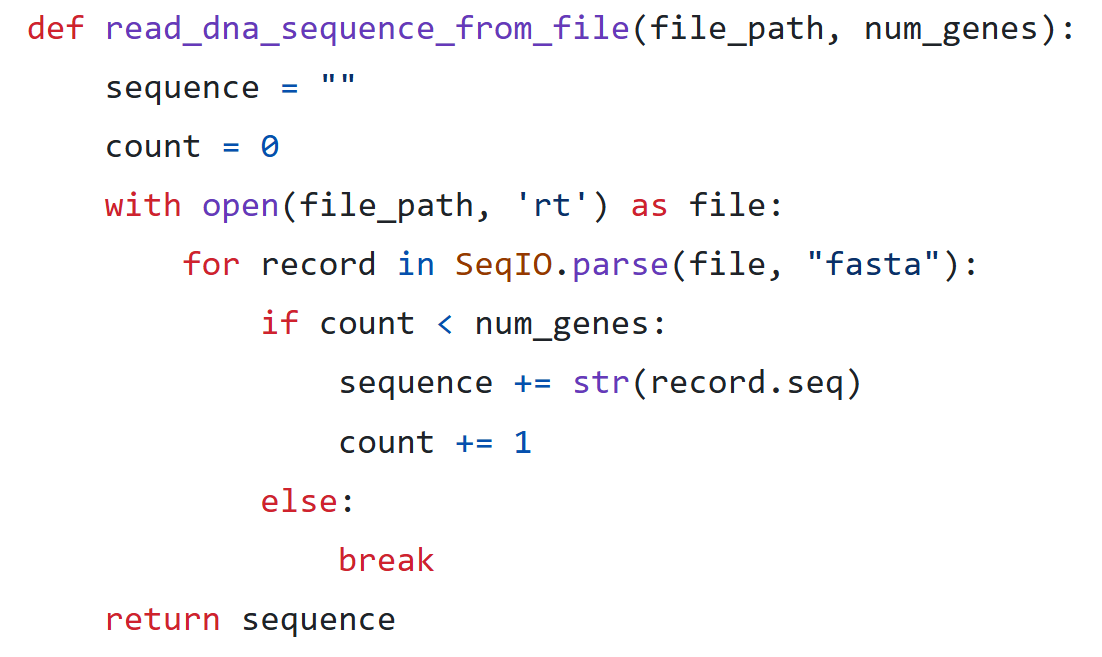
Identifying these inverted repeats can help in studying important functional regions within DNA structures, such as regulatory regions or protein binding sites. Additionally, the code improves processing efficiency through multiprocessing, making it suitable for large-scale DNA sequence analysis. It is designed to be flexible, capable of detecting both direct and spaced inverted repeats, making it suitable for studying diverse DNA structural features. Additionally, the modular structure of the code allows for easy adjustments, such as changing the minimum repeat length or spacer length, and can be extended to search for other DNA patterns, making it highly adaptable to different research needs.

The code utilizes several key libraries, including Biopython for reading and processing DNA sequences in FASTA format and generating reverse complements, gzip for directly reading compressed DNA files, and multiprocessing to speed up the analysis by dividing the DNA data into chunks and processing them in parallel across multiple CPU cores. To run the code, you need Python 3.x with Biopython installed, as well as a system with enough resources to handle large DNA datasets, such as multiple CPU cores and sufficient memory. The input files should be in FASTA or compressed .gz format, and the file paths must be correctly specified to ensure proper reading and saving of results.

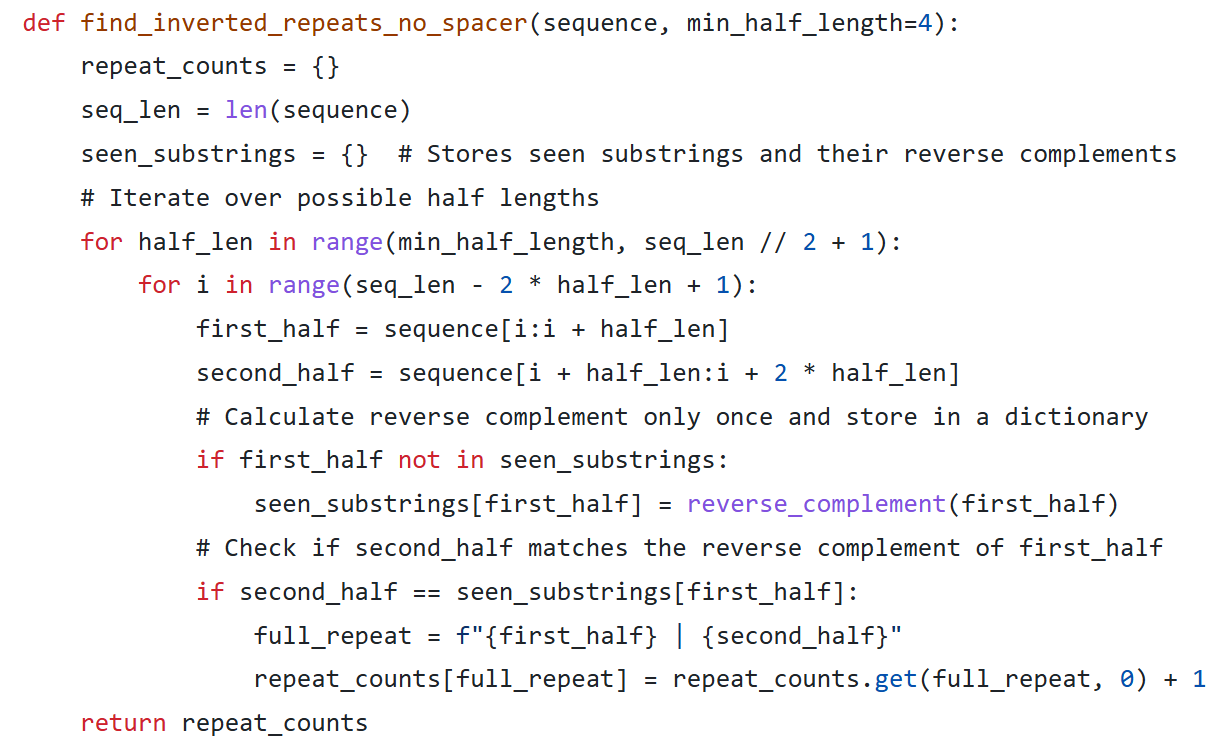
# The main analysis flow is as follows:

- Read DNA sequences from a FASTA format file with function read\_dna\_sequence\_from\_file(file\_path, num\_genes)

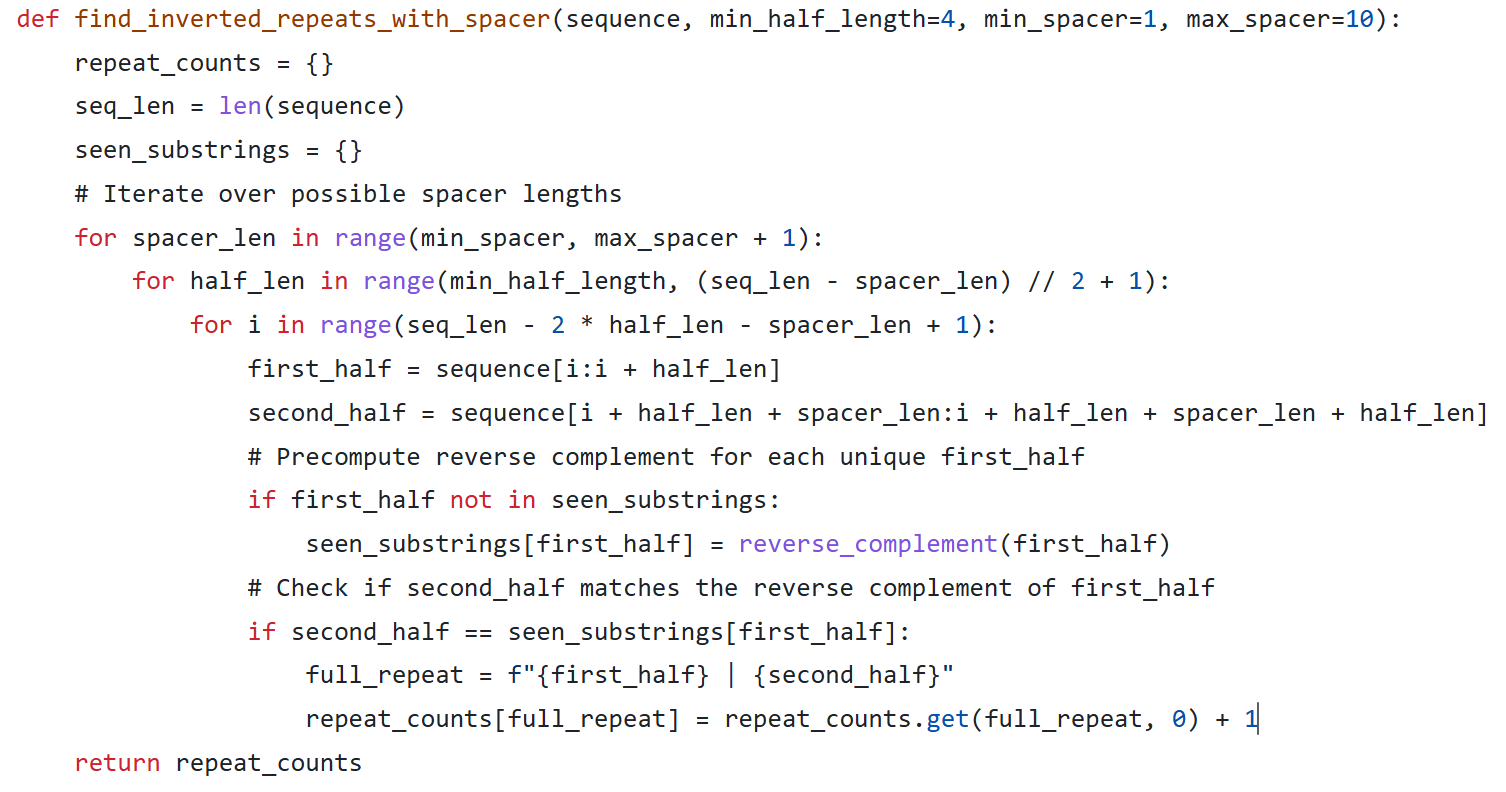
The input includes file\_path, which is the file path of the FASTA format file, and num-genes, which limits the number of genes read to shorten the time need to analyze.

- Find reversible repeats

The find\_inverted\_repeats\_no\_spacer function identifies inverted repeats in a DNA sequence where the two halves are directly adjacent. It iterates through possible lengths for each half of the repeat, calculates the reverse complement of the first half, and checks if it matches the second half. By storing computed reverse complements in a dictionary, the function avoids redundant calculations and efficiently counts the occurrences of each repeat pattern. This approach is suitable for identifying simple structural features in DNA, focusing on direct repeat pairs.



The find\_inverted\_repeats\_with\_spacer function, on the other hand, finds inverted repeats separated by a specified range of spacer lengths, with 1 to 10 bases. It similarly iterates through possible half lengths but also includes different spacer lengths between the two halves, making it more flexible for analyzing complex patterns. It uses a dictionary for precomputed reverse complements to improve performance, enabling the detection of spaced inverted repeats. This method is valuable for studying regions where repeats are separated by intervening bases, offering deeper insights into more complex DNA structures. Together, these functions provide a comprehensive toolset for analyzing both simple and complex patterns in DNA sequences.



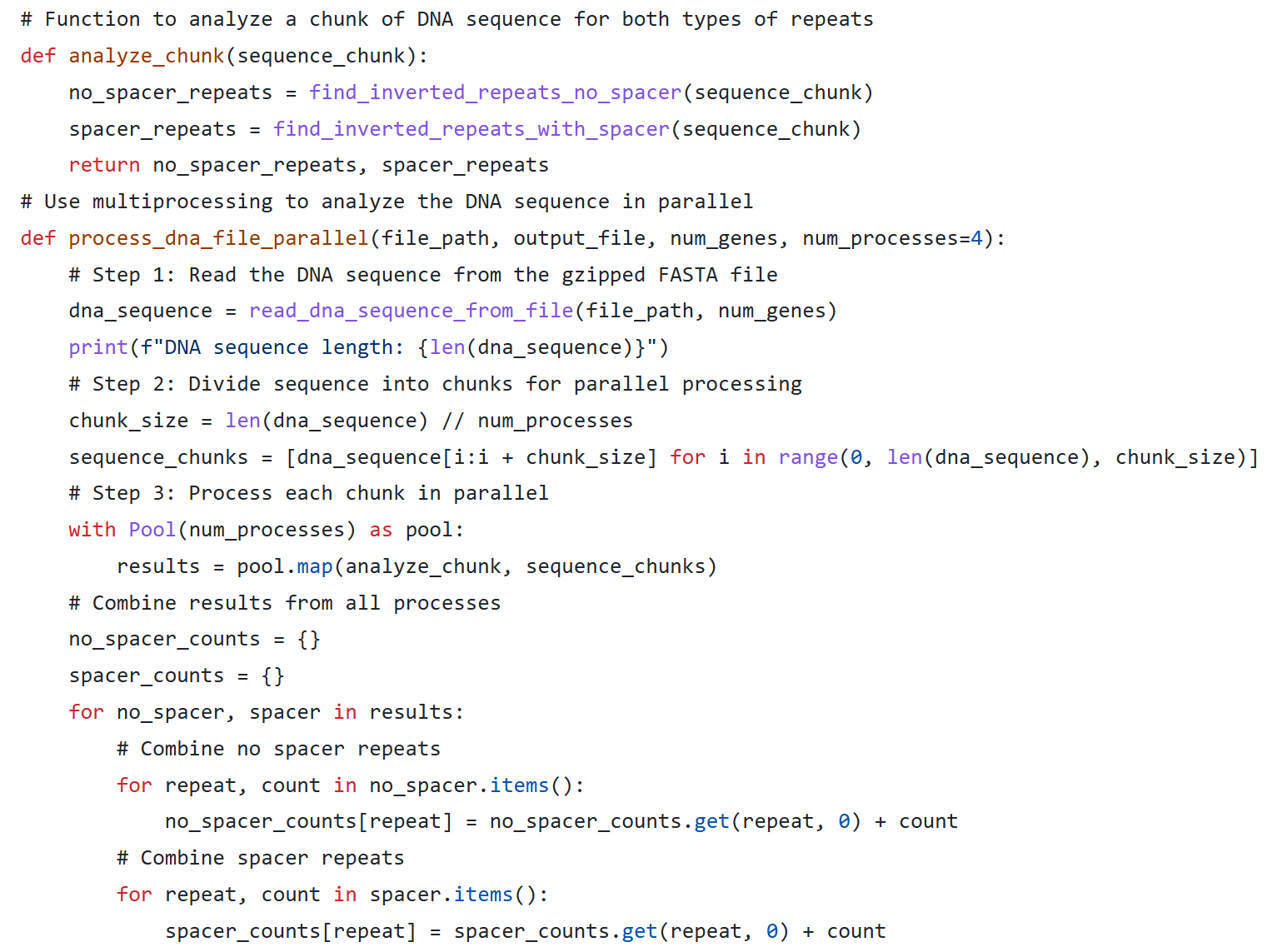
- Divide the DNA sequence into chunks, analyzing each chunk for inverted repeats, including those with and without spacers.

- Use multiprocessing to process each chunk in parallel and then combine the analysis results from all processes.

analyze\_chunk(sequence\_chunk)

process\_dna\_file\_parallel(file\_path, output\_file, num\_genes, num\_processes)

The process\_dna\_file\_parallel function has four key parameters: file\_path, output\_file, num\_genes, and num\_processes. file\_path specifies the location of the input DNA sequence file, while output\_file defines where the analysis results will be saved. num\_genes determines the number of gene sequences to read from the input file, allowing users to control the size of the data being analyzed. num\_processes sets the number of parallel processes used for the analysis, determining how many CPU cores will work simultaneously to process the DNA sequence in chunks, thus improving the speed and efficiency of the analysis.

This code processes DNA sequences by dividing them into smaller, manageable chunks and analyzing each for both types of inverted repeats—those with spacers and those without. By breaking the sequences into parts, it enables efficient parallel processing, which is especially beneficial when dealing with large datasets. Each chunk is analyzed simultaneously using multiple processors, significantly reducing the overall computation time. After processing, the results are combined to provide a complete analysis. This approach is ideal for large-scale genomic studies, as it optimizes processing speed and resource usage.

- Save the final results into an output file.

The output structure is designed to clearly separate and quantify the identified inverted repeats, both with and without spacers, allowing for straightforward analysis. It includes headers such as "Inverted repeats with no spacer" and "Inverted repeats with spacer," followed by detailed entries that list each repeat pattern along with its count and, for repeats with spacers, the spacer length. The purpose of this structure is to provide a comprehensive summary of the repeat occurrences, making it easy to load the data into analysis tools for further exploration, such as studying the frequency and distribution patterns of these repeats in the DNA sequence.

# How to Use the Code

To use this code, the user needs to provide the following parameters:

1. File path (`file\_path`): The path to the FASTA format file containing the DNA sequences to analyze.
2. Output file path (`output\_file`): The path for storing the analysis results in a text file.
3. Number of genes (`num\_genes`): The number of genes to read from the input file, which limits the sequence length.
4. Number of processes (`num\_processes`): The number of processes to use for parallel processing.

In summary, this code provides bioinformatics researchers with an efficient tool for analyzing specific patterns in DNA sequences, especially for computationally intensive tasks like identifying inverted repeats. The results can serve as foundational data for further research on gene functions and regulatory mechanisms.