This documentation explains the steps required to load a reference genome, find the first exon of the target gene (HPSE), extract its sequence, and save it for CRISPR guide design. The code utilizes **Biopython** for sequence parsing, **gzip** for compressed file handling, and **tqdm** for progress indication.

1. Loading the Necessary Libraries

We begin by importing the required libraries:

- `gzip`: To read compressed files.
- `SeqIO` **from Biopython**: To parse the FASTA file.
- `tqdm`: For progress indication while loading large files.

```
import gzip
from Bio import SeqIO
from tqdm import tqdm
```

2. Defining File Paths and Target Gene

The file paths for the genome FASTA file, the annotation GFF file, and the target gene (`HPSE`) are defined. The target gene is set based on the findings from Task 1.

```
# Define file paths
genome_fasta = "GCF_000001405.26_GRCh38_genomic.fna.gz"
annotation_gff = "GCF_000001405.26_GRCh38_genomic.gff.gz"
target_gene = "HPSE" # From task 1
```

3. Loading the Reference Genome

We load the entire reference genome from the compressed FASTA file using the `SeqIO` parser from Biopython. This function loads each chromosome's sequence into a dictionary.

```
# Function to load the reference genome
def load_reference_genome(genome_fasta):
    genome = {}
    with gzip.open(genome_fasta, "rt") as handle:
        for record in tqdm(SeqIO.parse(handle, "fasta"), desc="Loading genome"):
            genome[record.id] = record.seq
    return genome
```

4. Parsing the GFF File to Find the First Exon

This function scans the annotation GFF file for the target gene, extracts the chromosome location, strand information, and transcript/reference ID. Once the gene is found, the function returns the first exon coordinates.

```python
# Function to parse the GFF file, find the first exon of the gene, and return reference ID
def find_first_exon_location(gff_file, target_gene):
    exon_location = None
    found_gene = False
    reference_id = None  # Store reference/transcript ID
    with gzip.open(gff_file, "rt") as handle:
        for line in tqdm(handle, desc="Scanning GFF for first exon of target gene"):
            if line.startswith("#"):
                continue
            parts = line.strip().split("\t")
            if len(parts) < 9:
                continue
            feature_type = parts[2]
            attributes = parts[8]

            # Check for the gene
            if feature_type == "gene" and (f"Name={target_gene}" in attributes or f"gene_name={target_ger
                found_gene = True
                chrom = parts[0]
                strand = parts[6]

                # Extract the reference or transcript ID from attributes
                if "ID=" in attributes:
                    reference_id = attributes.split("ID=")[1].split(";")[0]
                elif "transcript_id=" in attributes:
                    reference_id = attributes.split("transcript_id=")[1].split(";")[0]
                else:
                    print("Reference ID not found in attributes.")

            # Look for the first exon of the gene
            if found_gene and feature_type == "exon":
                exon_start = int(parts[3])
                exon_end = int(parts[4])
                exon_location = (chrom, exon_start, exon_end, strand)
                print(f"First exon found from {exon_start} to {exon_end} on strand {strand}")
                break  # Stop after finding the first exon
    return exon_location, reference_id
```

5. Extracting the Gene Sequence

Once the exon coordinates are found, the next step is to extract the exon sequence from the genome. The function handles strand orientation (reverse-complement if necessary).

```python
# Function to extract the gene sequence from genome
def extract_gene_sequence(genome, chrom, start, end, strand):
    seq = genome[chrom][start:end]
    if strand == "-":
        seq = seq.reverse_complement()
    return seq
```

6. Running the Full Workflow

We now run the workflow by first loading the genome, locating the first exon of the gene **HPSE**, and extracting its sequence. The sequence is then saved in a FASTA file.

```python
# Load genome and find first exon for HPSE gene
genome = load_reference_genome(genome_fasta)
exon_location, reference_id = find_first_exon_location(annotation_gff, target_gene)

if exon_location and reference_id:
    chrom, start, end, strand = exon_location
    exon_seq = extract_gene_sequence(genome, chrom, start, end, strand)

    # Print the reference sequence ID and first exon sequence
    print(f"Reference sequence: {reference_id}")
    print(f"First exon sequence for {target_gene}:\n{exon_seq}\n")

    # Save the exon sequence in FASTA format, including the reference sequence ID in the header
    output_file = f"LPL_first_exon_{target_gene}_{chrom}.fasta"
    with open(output_file, "w") as f:
        f.write(f">{reference_id}_first_exon_{chrom}:{start}-{end}({strand})\n{exon_seq}\n")
    print(f"First exon sequence saved to {output_file}")
else:
    print(f"Gene {target_gene} not found or reference ID missing.")
```