Objective

This assignment involves processing the DNA sequence of the chromosome `chr1_GL383518v1_alt` (GRCh38.p13) to:

1. Read and extract specific nucleotide positions.
2. Generate the reverse complement of the sequence.
3. Count nucleotide occurrences per kilobase.
4. Summarize nucleotide distributions across the entire chromosome.

Instructions to Run

1. Open **Google Colab** or any Python environment.
2. Copy and run the provided Python script.
3. The script automatically:
    - Downloads the FASTA sequence for `chr1_GL383518v1_alt` from UCSC
    - Decompresses the `.fa.gz` file
    - Reads the sequence into memory
    - Performs all analysis as described in the assignment

Outputs

The script prints:

- **Specific nucleotide positions:** 10th and 758th bases
- **Reverse complement results:** 79th base and bases 500–800
- **Nucleotide counts per kilobase**
- **Lists of nucleotide counts for each kilobase**
- **Sum of nucleotides per kilobase** (expected ~1000; may vary due to ambiguous bases)

Steps

1. Downloading and Reading the FASTA File

- Downloads a compressed FASTA file (`.fa.gz`) directly from UCSC Genome Browser.

- Decompresses the file into a standard FASTA file (`.fa`).

- Reads the DNA sequence while skipping header lines (lines beginning with `>`).

2. Assignment Part 1: Print Specific Letters

- Prints the **10th nucleotide** in the sequence.

- Prints the **758th nucleotide** in the sequence.

3. Assignment Part 2: Reverse Complement

- Defines a function `reverse_complement_keep_case()` that:

  - Computes the reverse complement of the sequence.

  - Preserves letter case (uppercase and lowercase).

  - Skips ambiguous bases (`N` or `n`).

- Prints:

  - The **79th letter** of the reverse complement.

  - Bases from **500th through 800th positions** of the reverse complement.

4. Assignment Part 3: Nested Dictionary of Counts per Kilobase

- Defines a function `count_per_kb()` that:

- Splits the DNA sequence into **chunks of 1000 bases (kilobases)**.

- Counts **A, C, G, and T** in each chunk.

- Stores results in a nested dictionary:

```python
{
  0: {"A": ..., "C": ..., "G": ..., "T": ...},
  1000: {"A": ..., "C": ..., "G": ..., "T": ...},
  2000: {"A": ..., "C": ..., "G": ..., "T": ...},
  ...
```

}

```

**5. Assignment Part 4: Lists and Checks**

- (a) Creates a **list of 4 elements** `[A, C, G, T]` for the **first 1000 bases**.

- (b) Repeats this step for **each kilobase**.

- (c) Collects all these lists into a single **list of lists**.

- (d) Calculates the **sum of each list**:

  - Normally equals **1000**.

  - May be **less than 1000** if:

    - The last kilobase is shorter than 1000 bases.

  - The sequence contains ambiguous bases (`N`), which are not counted.

---

# Notes

- **Expected sum per kilobase:** 1000 bases
- **Possible discrepancies:**
    - The last kilobase has fewer than 1000 bases - 439 bases
    - Ambiguous nucleotides (N) are not counted in A/C/G/T
- **Environment:** Google Colab or any Python IDE with network access

---

# References


- UCSC Genome Browser: chr1_GL383518v1_alt sequence
- GRCh38.p13 Genome Reference Consortium


---
```