Prerequisite Terminologies:

In order to have a better understanding of the main topic, you should have the basic concept of the following terms:

- ➢ **Introduction to BioPython**
- ➢ **Bio.Seq Creating a Seq Object**

Introduction:

Bio.Alphabet is the class of BioPython which can be utilized to understand the alphabets that underlie within the sequences of interest. Alphabets used in Seq objects to declare sequence type and letters. This can be utilized for sequences which contain a finite number of similar words. Basically it allows us to understand the sequence type or whether there is ambiguity within the sequence of interest. Bio.Alphabet is the standard nucleotide and protein alphabets defined by IUPAC.

Steps:

- Import the 'Alphabet' module from the Bio.Alphabet class of BioPython package.

  ```
  from Bio.Seq import Seq
  from Bio.Alphabet import Alphabet
  ```

  [There are other alphabets also available such as generic_dna, we'll discuss that in the next section of this video. Here we'll learn to utilize the alphabet.]
- Declare the code that allows us to read the sequence from the user. Declare a variable (e.g: myDNA) and pass in the 'sequence'. Use the *print function* to print the given sequence.

  ```
  myDNA = "DNA Sequence"
  print(type(myDNA))
  print(myDNA)
  ```

  [This is basically a string variable.]
- Declare the code to convert it into a Seq object and use *type* and *print* functions to print sequence type.

  ```
  myDNA_seq_object = Seq(myDNA)
  print(type(myDNA_seq_object))
  ```

- Use the print function and pass in the seq object of the sequence along with the 'alphabet'.

print(myDNA_seq_object.alphabet)

[Here it'll not exactly print us the type of sequence but provide the simple 'Alphabet()' which means it can't figure out the sequence type.]

Note: The 'Alphabet class of BioPython' package will print the type of the seq object that what sort of sequence we've used but it is only possible when we already define the sequence alphabet in the declaration of seq object.

- Import a different module from the Bio.Alphabet class, let's say 'generic_dna'.

  from Bio.Alphabet import generic_dna

- Define the sequence alphabet in the declaration of seq object as:

  myDNA_seq_object = Seq(myDNA)
  print(type(myDNA_seq_object))
  myDNA_seq_object.alphabet = generic_dna
  print(myDNA_seq_object.alphabet)

  [Run the code, so it'll print the type of sequence as 'DNAAlphabet()' or whatever sequence you've used.]

- You can also define the sequence alphabet in the main declaration of the seq object. After providing the sequence to seq object or the function, we can define the alphabet here either like ***'alphabet = generic_dna'*** (whatever alphabet you would like it to be defined as) or you can simply type the alphabet type as ***'generic_dna'***.

  myDNA_seq_object = Seq(myDNA, generic_dna)
  print(type(myDNA_seq_object))
  print(myDNA_seq_object.alphabet)

  [Run the above code. You'll see the same results as earlier.]

Summary:

In this introductory video of Bio.Alphabet, we have learnt about the Bio.Alphabet class of BioPython package. We have also learnt how to utilize the Bio.Alphabet class to figure the alphabets that underlie within the sequences of interest.