```
seq[-1]
seq. (press tab key)
seq.find?
seq.find('ATG')
seq[42]
seq[42:45]
seq.count('ATG')
seq[::-1]
```

Python uses object oriented programming.

25 Example (Object Oriented Programming)
For example, consider the python command:

```
marker = MARKER(color = blue)
```

The function `MARKER(color = blue)` is a factory function which manufactures objects, in this case markers. The `color=blue` argument specifies that a blue marker should be manufactured.

Objects have attributes. For example, the color attribute `marker.color` should equal blue.

Objects also have methods. For example, the marker method `marker.change_color(red)` changes the color attribute of the marker from blue to red.

In IPython, typing `marker.` followed by the tab key will list all the attributes and methods associated with the marker object. Typing `marker?` will provide information about the marker object.

---

26 Example (Biopython)
Explain what the following Biopython commands do:

```
file = open('insulin_cDNA.txt')
seq = handle.readline().strip()
print seq

from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
DNA = Seq(seq,IUPAC.unambiguous_dna)
DNA
print DNA
DNA?
DNA. (press tab key)
print DNA.reverse_complement()
mRNA = DNA.transcribe()
print protein
protein.find('M')
protein.find('*')
print protein[14:125]
```

---

9 Lesson (Translating DNA)
Download the following files:

```
insulin_human_DNA.txt
insulin_human_cDNA.txt
```

(a) How many start codons are there in a the complete gene for human DNA? Make sure you check all six reading frames.

Solution:

(b) Translate the cDNA sequence for insulin to a protein sequence. Check your answer using Uniprot.

---

27 Example (Sequence Objects)
The first 10 letters of the human insulin protein sequence are: `MALWMRLLPL`. Lets use these letters to create a short sequence object using Biopython.

```
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
seq = Seq('MALWMRLLPL',IUPAC.protein)

seq
print seq
seq?
```

```
seq. (press tab)
seq.alphabet (sequence attribute)
seq.count('L') (sequence method)
```

---

28 Example (Fasta File Format)
The fasta file format is a common format for sequence data.

Single Sequence

```
>sp|P01308|INS_HUMAN Insulin OS=Homo sapiens GN=INS PE=1 SV=1
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAED
LQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
```

Multiple Sequences

```
>sp|P01308|INS_HUMAN Insulin OS=Homo sapiens GN=INS PE=1 SV=1
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAED
LQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
>sp|P67970|INS_CHICK Insulin OS=Gallus gallus GN=INS PE=1 SV=1
MALWIRSLPLLALLVFSGPGTSYAAANQHLCGSHLVEALYLVCGERGFFYSPKARRDVEQ
PLVSSPLRGEAGVLPFQQEEYEKVKRGIVEQCCHNTCSLYQLENYCN
>sp|P01310|INS_HORSE Insulin OS=Equus caballus GN=INS PE=1 SV=1
FVNQHLCGSHLVEALYLVCGERGFFYTPKAXXEAEDPQVGEVELGGGPGLGGLQPLALAG
PQQXXGIVEQCCTGICSLYQLENYCN
```

---

29 Example (Reading Single Sequence Fasta File)
Biopython can read fasta files.

```
from Bio import SeqIO
file = open('insulin_human.fasta')
seqrec = SeqIO.read(file,'fasta')
seqrec
print seqrec
seqrec?
seqrec. (press tab key)
print seqrec.id
print seqrec.seq
```

Biopython can also read fasta files containing multiple sequences. Instead of using the `SeqIO.read()` method, we use the `SeqIO.parse()` method for fasta files containing multiple sequences. The python `for` loop command, list comphrehension and the python `list` command are used with the `SeqIO.parse()` command to read fasta files with muliple sequences.

30 Example (Python For Loop)

```
animals = ['cow','sheep','goat']
for animal in animals:
    print animal
```

---

31 Example (Reading Multiple Sequence Fasta File)

```
from Bio import SeqIO
file = open('insulin_human_horse_chicken.fasta')
for seqrec in SeqIO.parse(file,'fasta'):
    print seqrec.id
    print seqrec.seq
```

---

32 Example (Python List Comprehension)

```
animals = ['cow','sheep','goat']
len('cow')
[len(animal) for animal in animals]
'cow'.count('e')
[animal.count('e') for animal in animals]
```

---

10 Lesson (Fasta Files)
Download the fasta file `insulin_76.fasta`.

(a) Print the sequence identifiers for all the sequences contained in this fasta file. (Hint: use a for loop.)

(b) Create a list of all the sequence identifies contained in this fasta file. Name the list `ids` for sequence ids. (Hint: use list comprehension.)

(c) How many sequences are in the `insulin_76.fasta` file. (Hint: use the `len()` command.)

---

33 Example (List of Sequence Records from Fasta File)
We can store all the sequences from a fasta file in a list using the `list` command.

```
from Bio import SeqIO
file = open('insulin_76.fasta')
seqrecs = list(SeqIO.parse(file,'fasta'))

seqrecs[0]
seqrecs[1]
seqrec[1].id
for i in range(len(seqrecs):
    print seqrecs[i]
    print
```

The SwissProt data format for protein sequence files contains more information than the fasta format.

34 Example (SwissProt Sequence Format)
Download the `insulin_76.txt` SwissProt sequence file and open it. Notice that it contains much more information than the corresponding `insulin_76.fasta` file.

```
from Bio import SwissProt
file = open('insulin_76.txt')
secrecs = list(SwissProt.parse(file))
rec0 = secrecs[0]
rec0. (press tab key)

for rec in seqrecs:
    print rec.accessions[0]
    print rec.organism
    print 'sequence length = %d' % rec.sequence_length
    print rec.sequence
    print
```

A small database of protein sequence information can be created and stored as a python dictionary

35 Example (Python Dictionary)
The words `'dog'`, `'cat'`, and `'mouse'` are the keys of the following Python dictionary:

```
amharic_dictionary = {'dog':'wosha','cat':'dimet','mouse':'eyet'}

amharic_dictionary['cat']
amharic_dictionary['mouse']
```

---

36 Example (Dictionary of Insulin Sequences)

```
from Bio import SeqIO
seqdict = SeqIO.to_dict(SeqIO.parse('insulin_76.txt','swiss')
seqdict.keys()
print seqdict['P01308']
```

---

37 Example (Web Download of Uniprot Files)
We can download sequence data directly from `uniprot.org` with the following commands:

```
from Bio import ExPASy
from Bio import SwissProt
web = ExPASy.get_sprot_raw('P01308')
seqrec = SwissProt.read(web)
print seqrec.accessions
print seqrec.entry_name
print seqrec.description
print seqrec.organism
print seqrec.seq
print
```

---

4 Sequence Alignment

DNA is subject to mutations. We will only consider insertions, deletions and substitutions.

38 Definition (Mutations)

```
orignal sequence   ATTGCTCC

orignal sequence   ATTG_CTCC
       insertion   ATTGGCTCC

orignal sequence   ATTGCTCC
        deletion   ATT_CTCC

orignal sequence   ATTGCTCC
    substitution   ATTTCTCC
```

39 Example (Sequence Alignment)
Consider the sequences:

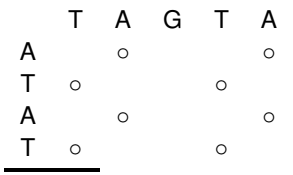
```
TAGTA
ATAT
```

Before we can determine how similar the sequences are to each other, we must first align the sequences. Two optimal alignments obtained using *dynamic programming* are:

```
TAGTA   _TAGTA
_A_TAT  ATA_T_
```

40 Example (Dot Plots)
Use a dot plot to compare the following sequences:

```
TAGTA
ATAT
```



11 Lesson (Dot Plots)
How similar are human, horse and chicken insulin? Use Jemboss to create dot plots comparing the insulin sequence for each.
- Go to www.uniprot.org.
- In the search field click on advanced.
- Select Gene name [GN] and type INS (for the insulin gene).
- Scroll down the results and click on the check box in the left column for human, horse and chicken insulin.
- Select download and a new window will appear containing the insulin sequences for human, horse and chicken in `fasta` format.
- Open Jemboss.
- Select ALIGNMENT, Dot Plots, polyplots.
- Cut and paste the `fasta` sequence data into Jemboss.
- Select pdf format for the output.
- Go to the Jemboss folder to retrieve the results.
- Interpret the plots.

12 Lesson (Dot Plots)
Repeat the previous lesson except compare the following insulin sequences:

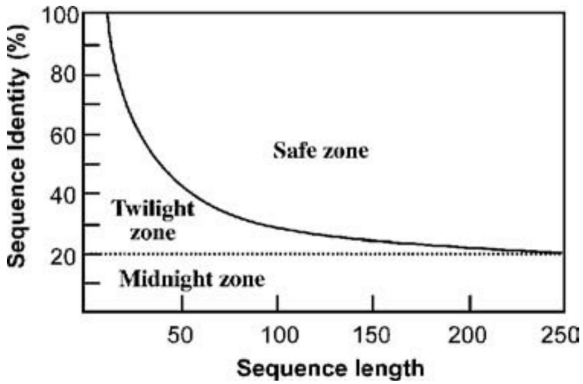
```
P01319 INS_CAPHI (Goat)
P01317 INS_BOVIN (Cow)
P01318 INS_SHEEP
```

(You may need to click in bottom right corner to display all insulin sequences at once.)

41 Definition (Homology)
Sequences which have evolved from a common ancestor are called **homologous**.

Similar sequences are likely to be homologous. However, we should keep in mind that sequences that have evolved from a distant ancestor may no longer be very similar to each other.

42 Definition (Sequence Alignment Zones)



Jin Xiong, Essential Bioinformatics, p. 33.

- safe zone: sequences are very likely to be homologous.
- twilight zone: sequences may be homologous.
- midnight zone: no reliable conclusion possible.

---

43 Definition (Percent Sequence Identity and Similarity)
After two sequences have been aligned, sequence identity and similarity is computed in one of two possible ways:

L_a is the length of the shorter sequence.

L_b is the length of the longer sequence.

N is either the number of identical or the number of similar letters in the alignment.

Sequence identity/similarity is computing using one of the two following formulas:

Formula 1

$$I = 100\,\frac{N}{L_a}$$

Formula 2

$$I = 100\,\frac{N}{\frac{L_a+L_b}{2}}$$

---

13 Lesson (Sequence Identity and Similarity)
Use `uniprot.org` to align cow insulin `P01317`, sheep insulin `P01318` and goat insulin `P01319`.

(a) In the `uniprot.org` search box type

`P01317 or P01318 or P01319`

Select the check boxes for these Ainsulin sequences and then select the alignment button. Wait a few seconds for the alignment to be computed by `uniprot.org`.

(b) Which sequences have a signal peptide attached? (Hint: check the box `signal peptide` in left column.)

(c) Which sequences have the propeptide attached? (Hint: check the box `propeptide` in left column.)

(d) Which sequences have the peptide segment? (Hint: check the box `peptide` in left column.)

(e) Complete the following tables *using only the peptide segment of each sequence*.

Sequence Identity:

	cow	sheep	goat
cow	100%		
sheep		100%	
goat			100%

Sequence Similarity:

	cow	sheep	goat
cow	100%		
sheep		100%	
goat			100%

Solution:

---

44 Definition (Paralogs)
If two sequences *from the same organism* are homologous, then the sequences are **paralogs**.

---

14 Lesson (Paralogs)
Use `uniprot.org` to align the insulin protein sequences: `P01325, P01326, P01322, P01323`.

(a) Which pairs of sequences are homologs and which are paralogs? Explain.

(b) Look at just the peptide segment of each sequence. (Check the box `peptide` in the left column.) Did the insulin gene duplicate before or after mouse and rat become separate species? Justify your answer.

---

45 Definition (Local vs Global Alignment)
Two basic types of sequence alignments are possible:

- local alignment (also called Smith-Waterman alignment)
- global alignment (also called Needleman-Wunsch alignment)

If sequence lengths are very different, we should consider using a local alignment. If the sequences are of similar lengths and likely to be closely related, we should use a global alignment. (Local alignments are used more often than global alignments.)

---

15 Lesson (Local Alignment)
Use Jemboss to align the cDNA and DNA sequences for human insulin. Identify where the exons and introns in the DNA sequence for human insulin are located.

---

16 Lesson (Phylogenetic Analysis)
Use `uniprot.org` to construct a phylogenetic tree for a group of several insulin sequences.

```
Go to uniprot.org
type gene:INS
select several interesting insulin sequences by clicking on the check box
align the sequences
select Tree under display
```

Discuss the results.

---

17 Lesson (BLAST)
Go to `uniprot.org`. Search for the entry `P01319` (goat insulin). Use the fast alignment algorithm called `BLAST` to find the most similiar entries to goat insulin from among all the many protein sequences in the database. Discuss your results. (Hint: click on the `BLAST` button.

Post Translational Modifications of Insulin

Once insulin has been translated from DNA into a protein sequence, it must be processed further before it can function as insulin. The unprocessed form of insulin is called *pre-proinsulin*.[1]

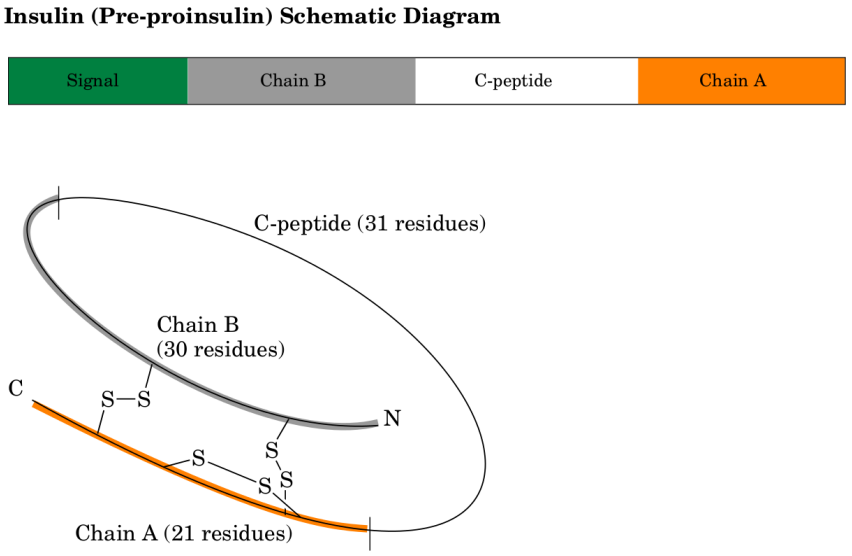


Figure 1: The post translational modification of pre-proinsulin.

First, the signal peptide tells the cell where insulin is to be transported. Then the signal peptide is removed. At this point the insulin is called *proinsulin*. The protein then folds with the help of the central C-peptide section. Three disulfied bonds form, two of which hold chain A and chain B together. The third disulfied bond is

[1] Mark Brandt, Chemistry and Biochemistry Dept., Rose-Hulman Insitute of Technology

within chain A. Finally the C-peptide is cut and removed. The final insulin product is only 51 amino acids long.

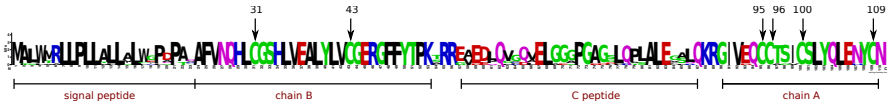


Figure 2: Sequence Logo of 11 aligned insulin sequences.

18 Lesson (Multiple Sequence Alignment)
Use the Ubuntu Software Center to install the multiple sequence alignment program `Clustal X`. Use this program to align all the insulin sequences in `insulin_76.fasta`. Discuss your alignment in the context of the post translational modification of insulin.