

Essential Computing for Bioinformatics

Lecture 4

High-level Programming with Python

Manipulating Files

MARC: Developing Bioinformatics Programs

July 2009

Alex Ropelewski

PSC-NRBSC

Bienvenido Vélez

UPR Mayaguez



- The following material is the result of a curriculum development effort to provide a set of courses to support bioinformatics efforts involving students from the biological sciences, computer science, and mathematics departments. They have been developed as a part of the NIH funded project “Assisting Bioinformatics Efforts at Minority Schools” (2T36 GM008789). The people involved with the curriculum development effort include:
- Dr. Hugh B. Nicholas, Dr. Troy Wymore, Mr. Alexander Ropelewski and Dr. David Deerfield II, National Resource for Biomedical Supercomputing, Pittsburgh Supercomputing Center, Carnegie Mellon University.
- Dr. Ricardo González Méndez, University of Puerto Rico Medical Sciences Campus.
- Dr. Alade Tokuta, North Carolina Central University.
- Dr. Jaime Seguel and Dr. Bienvenido Vélez, University of Puerto Rico at Mayagüez.
- Dr. Satish Bhalla, Johnson C. Smith University.
- Unless otherwise specified, all the information contained within is Copyrighted © by Carnegie Mellon University. Permission is granted for use, modify, and reproduce these materials for teaching purposes.
- Most recent versions of these presentations can be found at <http://marc.psc.edu/>

- Text Files
- Reading from Text Files
- Writing to Text Files
- Examples

What are Text Files?

- Persistent (non-volatile) storage of data
- Needed when:
 - data must outlive the execution of your program
 - data does not fit in memory (external algorithms)
 - data is supplied in batch form (non-interactive)
- Files are stored in your hard drive
- Files are maintained by your computer's Operating System (e.g. Linux, Windows, MacOS)

Examples of Text Files

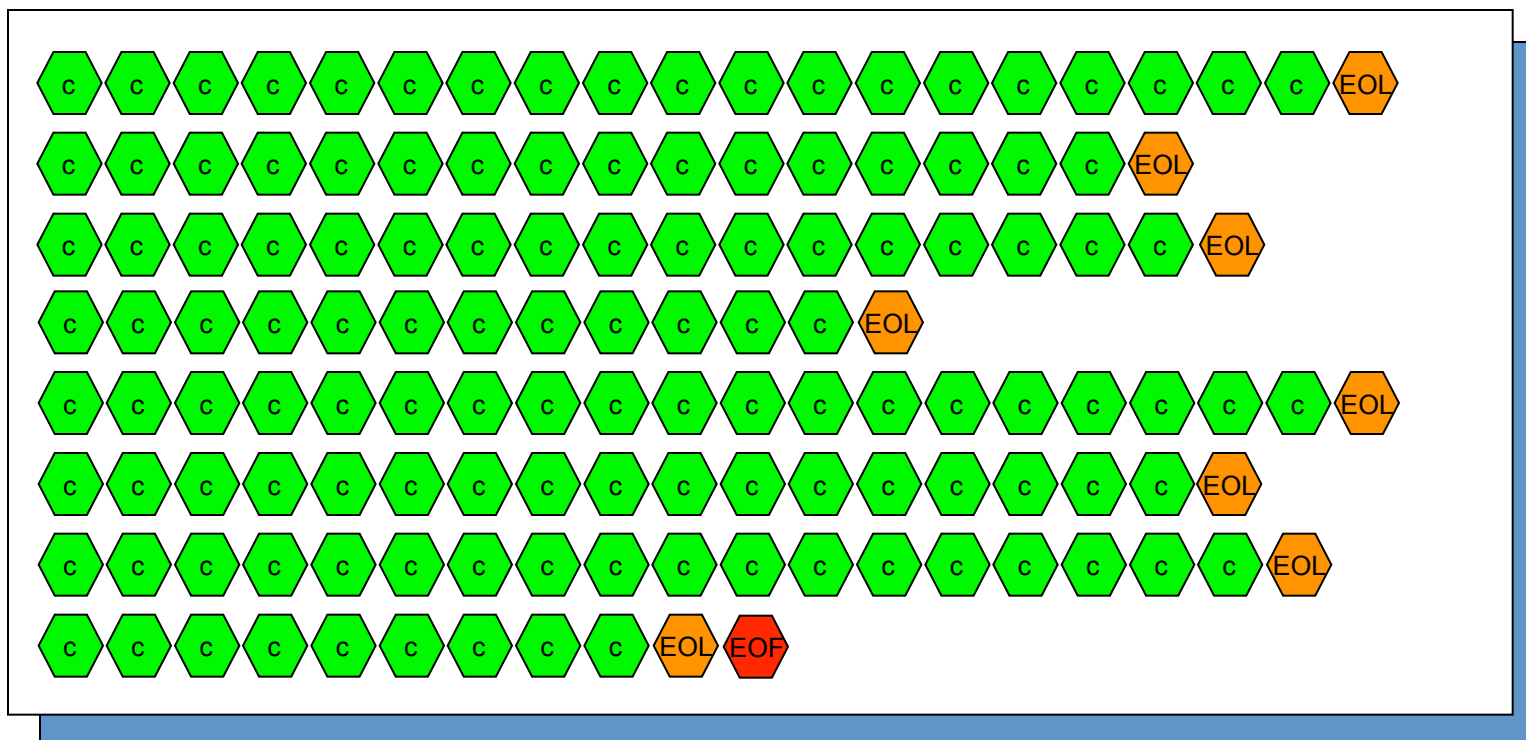
- Word documents
- Html documents retrieved from the web
- XML documents
- FASTA files
- GENBANK file
- Multiple Sequence Alignment Files (PFAM)

Text files contain a sequence of numbers that must be decoded using some standard in order to be converted to string form

Examples of encodings: ASCII, LATIN1, EBCDIC, Unicode

Check http://en.wikipedia.org/wiki/Character_encoding for more info

Top-level Anatomy of a Text File



Any Character



End-of-line Character



End-of-file Character

Invisible

Examples of Text Files: HTML

```
<html><head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <meta name="robots" content="index, nofollow, noarchive"/>
  <title>NCBI Sequence Viewer v2.0</title>
  <!--MUTABLE-->
  <!--www.ncbi.nlm.nih.gov:80-->
  <!--MUTABLE-->
  <link type="text/css" rel="stylesheet" href="http://www.ncbi.nlm.nih.gov/corehtml/ncbi_test.css"/>
  <link type="text/css" rel="stylesheet" href="../svviewer/viewer.css"/>
  <script type="text/javascript" src="../svviewer/viewer.js"> </script>

  <script type="text/javascript" src="http://www.ncbi.nlm.nih.gov/coreweb/javascript/popupmenu2/popupmenu2_6loader.js"> </script>
</head><body ><form name="frmQueryBox0" action="/sites/entrez" method="get" style="margin:0">
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td>
      <table width="100%" border="0" cellpadding="0" cellspacing="0">
        <tr>
          <td align="left" width="130">
            <a href="http://www.ncbi.nlm.nih.gov">

              
            </a>
          </td>
          <td align="left">
            
          </td>
          <td>
            <table class="medium1" border="0" bordercolor="#336699" cellpadding="2" cellspacing="0" align="right">
              <tr>
                ...
                ...
              </tr>
            </table>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
</body>
</html>
```

Examples of Text Files: XML

```
<?xml version="1.0"?>
<!DOCTYPE Seq-entry PUBLIC "-//NCBI//NCBI Seqset/EN" "http://www.ncbi.nlm.nih.gov/dtd/NCBI_Seqset.dtd">
<Seq-entry>
  <Seq-entry_set>
    <Bioseq-set>
      <Bioseq-set_level>1</Bioseq-set_level>
      <Bioseq-set_class value="nuc-prot"/>
      <Bioseq-set_descr>
        <Seq-descr>
          <Seqdesc>
            <Seqdesc_source>
              <BioSource>
                <BioSource_genome value="genomic">1</BioSource_genome>
                <BioSource_org>
                  <Org-ref>
                    <Org-ref_taxname>Xanthomonas campestris pv. campestris</Org-ref_taxname>
                    <Org-ref_db>
                      <Dbtag>
                        <Dbtag_db>taxon</Dbtag_db>
                        <Dbtag_tag>
                          <Object-id>
                            <Object-id_id>340</Object-id_id>
                          </Object-id>
                        </Dbtag_tag>
                      </Dbtag>
                    </Org-ref_db>
                    <Org-ref_orgname>
                      <OrgName>
                        <OrgName_name>
                          <OrgName_name_binomial>
                            ...
                          ...
                        ...
                      ...
                    ...
                  ...
                ...
              ...
            ...
          ...
        ...
      ...
    ...
  ...
</Seq-entry_set>
</Bioseq-set>
</Seq-entry>
```

Examples of Text Files: GENE BANK

```

LOCUS      NC_010688                5079002 bp    DNA        circular BCT 17-JUL-2008
DEFINITION Xanthomonas campestris pv. campestris, complete genome.
ACCESSION  NC_010688
VERSION    NC_010688.1  GI:188989396
PROJECT    GenomeProject:29801
KEYWORDS   complete genome.
SOURCE     Xanthomonas campestris pv. campestris
  ORGANISM Xanthomonas campestris pv. campestris
            Bacteria; Proteobacteria; Gammaproteobacteria; Xanthomonadales;
            Xanthomonadaceae; Xanthomonas.
REFERENCE  1
  AUTHORS  Vorholter,F.J., Schneiker,S., Goesmann,A., Krause,L., Bekel,T.,
            Kaiser,O., Linke,B., Patschkowski,T., Ruckert,C., Schmid,J.,
            Sidhu,V.K., Sieber,V., Tauch,A., Watt,S.A., Weisshaar,B.,
            Becker,A., Niehaus,K. and Puhler,A.
  TITLE    The genome of Xanthomonas campestris pv. campestris B100 and its
            use for the reconstruction of metabolic pathways involved in
            xanthan biosynthesis
  JOURNAL  J. Biotechnol. 134 (1-2), 33-45 (2008)
  PUBMED   18304669
REFERENCE  2 (bases 1 to 5079002)
  CONSRM   NCBI Genome Project
  TITLE    Direct Submission
  JOURNAL  Submitted (22-MAY-2008) National Center for Biotechnology
            Information, NIH, Bethesda, MD 20894, USA
REFERENCE  3 (bases 1 to 5079002)
  AUTHORS  Linke,B.
  TITLE    Direct Submission
  JOURNAL  Submitted (03-DEC-2007) Linke B., Center For Biotechnology,
            Bielefeld University, Universitaetsstrasse 25, 33501 Bielefeld,
            GERMANY
COMMENT    PROVISIONAL REFSEQ: This record has not yet been subject to final
  ...

```

Examples of Text Files: FASTA

```
>gi|188989396|ref|NC_010688.1| Xanthomonas campestris pv. campestris, complete genome
ATGGATGCTTGGCCCCGCTGTCTGGAACGTCTCGAAGCTGAATTCCTCGCCCGAAGATGTCCACACCTGGT
TGAAACCCCTGCAGGCCGAAGATCGCGGCGACAGCATCGTGCTGTACGCGCCGAACGCCCTTCATTGTCTGA
GCAGGTTTCGCGAGCGATACCTGCCGCGCATCCGCGAGTTGTGGCGTATTTGCGCCGGCAATGGCGAGGTG
GCGCTGGCGGTCGGCTCCCGTCCGCGTGCGCCGGAGCCGCTGCCGGCACCGCAAGCCGTCGCCAGTGCGC
CGGCGGCCGCGCCGATCGTGCCCTTCGCCGGCAACCTGGATTTCGATTACACCTTTGCCAACTTCGTGGA
AGGCCGAGCAACCAGCTCGGTCTGGCCGCGGCGATCCAGGCCGCACAGAAGCCTGGCGACCGGGCGCAC
AACCCGTTGCTGTGTACGGCAGCACCGGGCTGGGCAAGACCCACCTGATGTTGCGCGCCGGCAACGCGC
TGCGCCAGGCCAATCCGGCCGCCAAGGTGATGTACCTGCGCTCGGAACAGTTCTTCAGCGCGATGATCCG
CGCGTTGCAGGACAAGGCAATGGACCAGTTCAAGCGCCAGTTCCAGCAGATCGATGCGCTGCTGATCGAC
GACATCCAGTTTTTTTGGCCGCAAGGACCGCACGCGAGGAGGATTTTTTCCACACCTTCAACGCGCTGTTTCG
ACGGCCGCCAGCAGATCATCTGACCTGCGACCGCTATCCGCGCGAAGTCGAGGGCCTGGAGCCGCGGCT
GAAGTCGCGCCTGGCCTGGGGCCTGTCGGTGGCGATCGACCCGCCGATTTTCGAGACGCGTGCGGCAATC
GTGCTGGCCAAGGCGCGGAGCGCGGCGCCGAGATTCCCGACGACGTGGCGTTTCTGATCGCCAAGAAGA
TGCGCTCGAACGTGCGCGACCTGGAAGGGGCGCTCAACACGTTGGTGGCCCGCGCCAACCTCACTGGCCG
TTCGATCACCGTGGAGTTTTCGCGAGGAGACGCTGCGTGACCTGTTGCGTGCGCAGCAGCAGGCGATCGGC
ATTCCTCAACATCCAGAAGACCGTGGCCGACTACTACGGCCTGCAGATGAAGGACCTGCTTTTCAAGCGCC
GCACCCGCTCATTTGGCGCGCCCGCGCCAGGTGGCGATGGCGCTCGCCAAGGAGTTGACCGAGCACAGCCT
TCCCGAGATCGGCGATGCGTTTGCCGGCCGCGACACACACCGTGCTGCACGCTGCCGGCAGATCCGC
ACGCTGATGGAGGCCGACGGCAAGCTGCGCGAGGACTGGGAAAAGCTGATTCGCAAGCTCAGCGAGTGAG
CGTTTGGCGCCCTTCTCGCCCGGCTTGGAAGGAAAGCTGGATGAGTTGGGGCCAAATGCGGGAGAATCTG
TGGATAAATGCGGCTGCAGCGTTGAGGCGGAACTATCCACAGGTTTTCCACCGCTCCAGGGGCACTAG
TCCATAGACTTTGAAGCTCAAAATGGTCTTTGGAAACAATACGTTAGTGTTGTTGTCCGCCGAAAACGGC
CCTACCATCACCAAGCTTTTGATTTATTCCACATCTTTAAAGCATAGGGGCGACGGAACCACATGCGT
TTCACACTGCAGCGCGAAGCCTTCCTCAAGCCGTTGGCCAGGTGGTCAATGTCGTGCAACGGCGTCAGA
CATTGCCGGTACTGGCGAAGCTTGCTGGTGAGGTGAACAACGGCCAGCTGTCGCTGACGGGGACCGACCT
GGAAGTCGAAATGATCTCGCGACCATGGTTCGAGGACGCCCAGGACGGCGAAACCACGATCCCGGCGCGC
AAGCTGTTTCGACATCCTGCGGGCCCTGCTGACGGCAGCCGTGTCACCGTCTCGCAAACCGGAGACAAGG
TCACGGTGCAGGCCGGGCGCAGCCGCTTTACGCTCGCCACGCTACCGGCCAACGACTTCCCGTCGGTGGA
CGAAGTCGAGGCCACCGAGCGTGTGGCGGTGCCGGAAGCCGGGCTGAAGGAGCTGATGGAGCGCACGGCG
TTTCGCCATGGCCCGAGCAGGACGTGCGTTATTACCTCAACGGCCTGCTGTTTCGACCTGCGCGATGGCCTGC
```

...

...

Examples of Text Files: ClustalW

clustalw

```

0xOLI      RIILVTGASDGIGREAAPTYARY--GATVILLGRN-----EEKLRQV--
1xMAN      KKVIVTGASKGIGREMAHYHLAKM-GA-HVVVTARS-----KETLQKV--
2xAST      KVILITGASRGIGLQLVKTVEEDDECIVYGVRT-----EAGLQSL--
3xAST      KVILVTGVSRGIGKSIVDVLFLSKDTVVYGVARS-----EAPL--
4xAST      KIAVVTGASGGIGYEVTKELARN--GYLVYACARR-----LEPMAQL--
5xMAN      HVALVTGGNKGIGLAIVRDLCLRL-FSGDVVLTARD-----VTRGQAAV--
6xCSU      NTVLITGGSAGIGLELAKRLEL--GNEVIICGRS-----EARLAEAK--
7xEX       KTVIITGGARGLGAEAAQAVAA-GARVVLADVLD-----E-EGAATA--
8xASP      KTVLLTGASRGLGVYIARALAKE--QATVVCVSRS-----QSGLAQT--
9xCSU      KTALITGGGRGIGRATALALAKE--GVNIGLIGRT-----SANVEKV--
10xOBR     KIALVTGAMGGLGTAICQALAKD-GCIVAANCLPN-----FEPAAAWL--

```

```

0xOLI      ---ASHIN--EETG-RQPQWFILDLLTCTSENC-QQLAQRIAVNY----P-RLDGVVLHNA
1xMAN      ---VSHCL---ELG-AASAHYIA-GT---MEDM-TFAEQFVAQAG--KLMGGLDMLILNH
2xAST      ---QREYG-----ADKFVYRVLD---ITDR-SRMEALVEEIR--QKHGKLDGIVANA
3xAST      ---KKLK--EKYG-DRFFYVVG--D---ITED-SVLKQLVNAAVK--GHGKIDSLVANA
4xAST      ---AIQ----FG-NDSIKPYK-LD---ISKP-EEIVTFSGFLRANLPDGKLDLLYNNA
5xMAN      ---QQLQ---AEG--LSPRFHQ-LD---IDDL-QSIRALRDFLR--KEYGGLDVLVNNA
6xCSU      ---QQLP-----N-IHTKQ-CD---VADR-SQREALYEWALK--EYPNLNVLVNNA
7xEX       ---R---ELG--DAARYQH-LD---VTIE-EDWQRVVAYAR--EEFGSVDGLVNNA
8xASP      ---CNAVK---AAG--GKAIAIP-FD---VRNT-SQLSALVQQAQ--DIVGPIDVLINNA
9xCSU      ---AEEVK---ALG--VKAAFAA-AD---VKDA-DQVNQAVAQVK--EQLGDIDILINNA
10xOBR     ---GQQE---ALG--FKFYVAE-GD---VSDF-ESCKAMVAKIEA--DLGPVDILVNNA

```

...

...

Loading Fasta Sequences: Step 0

```
# ex. 1
#
# open a file for reading
# read a single line of the file and
# bind the line to variable 'line'
#

file = open("test.txt", "r") # Open the file
line = file.readline()      # Returns next line in a string
print line
file.close()                # Close the file
```

How can we scan the entire sequence of lines?

Loading Fasta Sequences: Step 1

```
# ex. 3
#
# read all the lines of the file and store them in a list
# print the 3rd line
#

file = open(pathname, "r")
line = file.readlines() # Reads all lines into a sequence
print line[2]
file.close()
```

Why is this NOT a great idea?

How can we do better?

Loading Fasta Sequences: Step 2

```
# ex. 4
#
# read 1 line at a time
# print the line number and the line
#

def example4(pathname):
    file = open(pathname, "r")
    count = 0
    for line in file:
        print count, ":", line
        count = count + 1
    file.close()
```

Now lets try to extract info from the Fasta file?

A Fasta file with three sequences

```
>gi|188989396|ref|NC_010688.1| Xanthomonas campestris pv. campestris, complete genome
ATGGATGCTTGGCCCCGCTGTCTGGAACGTCTCGAAGCTGAATTCCCGCCCCGAAGATGTCCACACCTGGT
TGAAACCCCTGCAGGCCGAAGATCGCGGCGACAGCATCGTGCTGTACGCGCCGAACGCCCTTCATTGTCTGA
GCAGGTTTCGCGAGCGATACCTGCCGCGCATCCGCGAGTTGCTGGCGTATTTTCGCCGGCAATGGCGAGGTG

>gi|194208364|ref|XM_001500342.2| PREDICTED: Equus caballus electron-transferring-flavoprotein ...
GATACGTGACCCGGAAGCCTCTGCTGGCCATGGCGTCATGCGTGGCGCCGGCGCAGAGCGAGAGAGAGTC
GGGAGCGCTGTGAAGACAGAGCGGTCGGCTGATCAGAGACGAACTTCAGTGGAGGTGATGGCGCCCCCGG
CGGCCTAGAGGTCCAGAGCGTGCCGCGAGCTGCAGACAGTACGCCCTCCCATTTGTATCCGACGGAGACTCC
TCGTTGCAGGGAACATGTTGCTGCCGCTAGCCAAGCTGTCCTGTCCGGCATATCAGTGCTTTTCATGCCTT
AAAAATTAAGAAAGATTATCTACCTCTGTGTGCTACAAGATGGTCTTCAACTTCTGTTGTACCTCGAATT

>gi|193087197|gb|CP001100.1| Chloroherpeton thalassium ATCC 35110, complete genome
ATGCTTATGAGCGAAGGACATGACCAGCCAGGCGCTCCTGTTTCTCATTTATCGGAACAGGCTTTAGCAC
AAATTGCGTGGAAAAATGCCTCGATATCATTCGTGATGGCCTTCATAACCTGCAAAGCTTTAAGACTTG
GTTTGAGCCCATCGTGCCATTAAACTTTCTGGCCAGGAATTGACCATTGAGGTGCCAGCCAGTTTTTTT
TATGAAATGATCGAGGAAAAATTATTACTCGCTTTTAAAGCGCGCCTTGATGGAAGTGATGGGAACGGGAG
CCAAGCTGCGATATTCTGTTTTGGTTTCAGGCGGCACAAGCTGAAACACCCGTCGTCGCTCGTATCCCCGA
GAAAAAGGCAAGCACACTCCGGCGGCGCTTCCGAAAGTCATTTCTCATGCGAATGGCAAGCAGACAAAA
GAAGCTCAAGACTTATTTGCAAACAATGTACACCGCTTCGAAAGCTACCTAAATCCAAAATATCGCTTCG
```

Loading Fasta Sequences: Step 3

```
# ex. 5
#
# read 1 line at a time
# print only the fasta sequence header
#

def example5(pathname):
    file = open(pathname,"r")
    count = 0
    for line in file:
        if line[0] == '>':
            print count,":",line
            count = count + 1
    file.close()
```

Now lets try to extract info from the Fasta file?

Loading Fasta Sequences: Step 4

```
# ex. 6
#
# read 1 line at a time
# identify the fasta sequence header and the sequence data
#

def example6(pathname):
    file = open(pathname,"r")
    for line in file:
        if line[0] == '>':
            print "header",line
        else:
            print "data",line
    file.close()
```

How about blank lines and trailing newlines?

Loading Fasta Sequences: Step 5

```
# ex. 8
#
# read 1 line at a time
# identify the fasta sequence header and the sequence data
# account for blank lines and trailing space/newlines
# collect sequence data
def example8(pathname):
    file = open(pathname,"r")
    data = ""
    for line in file:
        # remove the trailing '\n' and trailing spaces
        line = line.rstrip('\n ')
        # if the line length is < 1, there nothing to do for this line
        # so move to the next line
        if len( line ) < 1:
            continue
        if line[0] == '>':
            print "data",data
            data = ""
            print "header",line
        else:
            data = data + line
    file.close()
```

How about blank lines and trailing
newlines?

Loading Fasta Sequences: Step 6

```
def readFastaFile(filename):  
    file = open(filename,"r")  
    sequence_data = []  
    for line in file:  
        # remove the trailing '\n' and trailing spaces  
        line = line.rstrip('\n ')  
        # if the line length is < 1, do nothing  
        # so skip rest of iteration  
        if len( line ) < 1:  
            continue  
        if line[0] == '>':  
            sequence_data.append('')  
        else:  
            k = len(sequence_data) - 1  
            sequence_data[k] = sequence_data[k] + line  
    file.close()  
    return(sequence_data)
```

How about blank lines and trailing newlines?

Summary of File Operations

Method	Action
Read([n])	Reads at most n bytes; if no n is specified, reads the entire file
Readline([n])	Reads a line of input, if n is specified reads at most n bytes
Readlines()	Reads all lines and returns them in a list
Xreadlines()	Reads all lines but handles them as a xrange type
Write(s)	Writes string s
Writelines(l)	Writes all strings in list l as lines
Close()	Closes the file
Seek(offset[, mode])	Changes to a new file position=start + offset. Start is specified by the mode argument: mode=0 (default), start = start of the file, mode=1, start = current file position and mode=2, start = end of the file.

Importing ClustalW Files

clustalw

```

0xOLI      RIILVTGASDGIGREAAPTYARY--GATVILLGRN-----EEKLRQV--
1xMAN      KKVIVTGASKGIGREMAHYHLAKM-GA-HVVVTARS-----KETLQKV--
2xAST      KVILITGASRGIGLQLVKTVEEDDECIVYGVART-----EAGLQSL--
3xAST      KVILVTGVSRGIGKSIVDVLFLSLDKDTVYGVARS-----EAPL--
4xAST      KIAVVTGASGGIGYEVTKELARN--GYLVYACARR-----LEPMAQL--
5xMAN      HVALVTGGNKGIGLAIVRDLCLRL-FSGDVVLTARD-----VTRGQAAV--
6xCSU      NTVLITGGSAGIGLELAKRLEL--GNEVIICGRS-----EARLAEAK--
7xEX       KTVIITGGARGLGAEAAQAVAA-GARVVLDVLD-----E-EGAATA--
8xASP      KTVLLTGASRGLGVYIARALAKE--QATVVCVSRS-----QSGLAQT--
9xCSU      KTALITGGGRGIGRATALALAKE--GVNIGLIGRT-----SANVEKV--
10xOBR     KIALVTGAMGGLGTAICQALAKD-GCIVAANCLPN-----FEPAAAWL--

```

```

0xOLI      ---ASHIN--EETG-RQPQWFILDLLTCTSENC-QQLAQRIAVNY----P-RLDGVVLHNA
1xMAN      ---VSHCL---ELG-AASAHYIA-GT---MEDM-TFAEQFVAQAG--KLMGGLDMLILNH
2xAST      ---QREYG-----ADKFVYRVLD---ITDR-SRMEALVEEIR--QKHGKLDGIVANA
3xAST      ---KKLK--EKYG-DRFFYVVG--D---ITED-SVLKQLVNAAVK--GHGKIDSLVANA
4xAST      ---AIQ----FG-NDSIKPYK-LD---ISKP-EEIVTFSGFLRANLPDGKLDLLYNNA
5xMAN      ---QQLQ---AEG--LSPRFHQ-LD---IDDL-QSIRALRDFLR--KEYGGLDVLVNNA
6xCSU      ---QQLP-----N-IHTKQ-CD---VADR-SQREALYEWALK--EYPNLNVLVNNA
7xEX       ---R---ELG--DAARYQH-LD---VTIE-EDWQRVVAYAR--EEFGSVDGLVNNA
8xASP      ---CNAVK---AAG--GKAIAIP-FD---VRNT-SQLSALVQQAQ--DIVGPIDVLINNA
9xCSU      ---AEEVK---ALG--VKAAFAA-AD---VKDA-DQVNQAVAQVK--EQLGDIDILINNA
10xOBR     ---GQQE---ALG--FKFYVAE-GD---VSDF-ESCKAMVAKIEA--DLGPVDILVNNA

```

...

...

Reading from ClustalW Files

```
def loadClustalwAlignment(pathname):
    sequences = []
    id2SequenceMap = dict() # Keep id -> seq dictionary to collect data
    f=open(pathname, 'r')
    f.readline()
    for line in f:
        line = line.replace('\n','')
        if (len(line)>1):
            sequenceID = extractID(line)
            if (sequenceID in id2SequenceMap):
                prevSequence = id2SequenceMap[sequenceID]
            else:
                prevSequence = ' '
            id2SequenceMap[sequenceID]= prevSequence + extractSequence(line)
    f.close()
    sequences = []
    for key in id2SequenceMap.keys():
        sequences = sequences + [key + ":" + id2SequenceMap[key]]
    return sequences
```

Writing to Text Files

```
def translateFastFile(infilename, outfilename):  
    infile = open(filename, "r")  
    outfile = open(filename, "w")  
    sequence_data = []  
    for line in infile:  
        # remove the trailing '\n' and trailing spaces  
        line = line.rstrip('\n ')  
        # if the line length is < 1, do nothing  
        # so skip rest of iteration  
        if len( line ) < 1:  
            continue  
        if line[0] == '>':  
  
            sequence_data.append('')  
        else:  
            k = len(sequence_data) - 1  
            sequence_data[k] = sequence_data[k] + line  
    file.close()  
    return(sequence_data)
```



Homework

