Course Code: BT 3172

Course Title: Special Topics in Bioinformatics

BT-3172: Special Topics in Bioinformatics
Lab 13: Introduction to the Unix shell for biological applications.

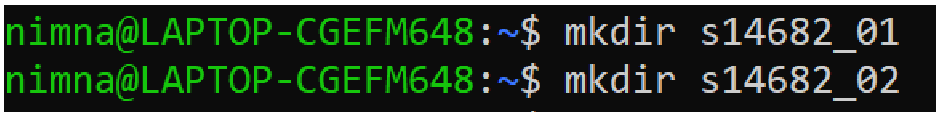
Name : Nimna Alupotha Gamage (NIMNA A. G. T.)

Index No.: s14682

Reg. No. : 2019s17241

In this practical, you will learn how to use basic Unix shell commands to solve simple biological questions.

Create 2 folders for the two questions and name them in “Your_index_question_no” format.



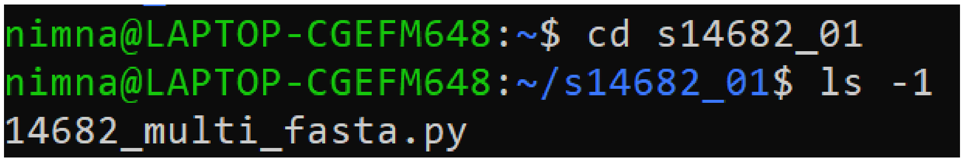
```
nimna@LAPTOP-CGEFM648:~$ mkdir s14682_01
nimna@LAPTOP-CGEFM648:~$ mkdir s14682_02
```

Use the Unix shell/command line to implement your commands. Follow the question-specific instructions and save the necessary Python and shell script files in their folders. Also, make sure all the outputs specified in the questions are in the folders. Finally, compress the folders and upload them to the LMS. **You must write the Unix shell commands and Python scripts in the space below each question for evaluation.**

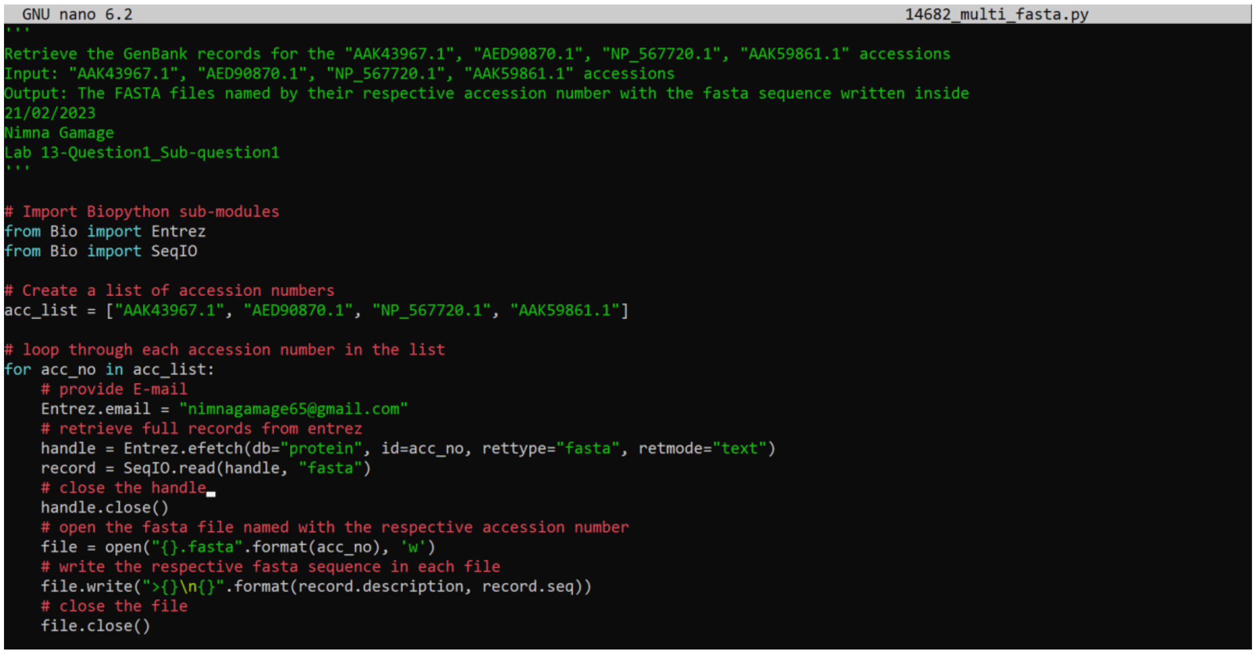
1) Processing multiple FASTA files using the Unix shell.

   In this problem, you will be working with APETALA2/ETHYLENE RESPONSIVE FACTOR (AP2/ERF) family transcription factors (Zhouli, et al., 2019) which contain the AP2/ERF DNA-binding domain.

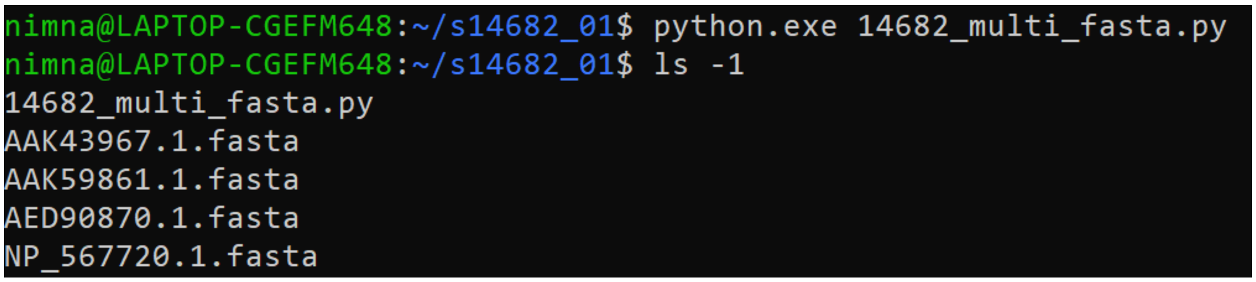
   I. Write a Python script to retrieve the GenBank records for the following accessions: "AAK43967.1","AED90870.1","NP_567720.1", and "AAK59861.1", and save their amino acid sequences in **separate** FASTA files. The FASTA files can be named by their respective accession number. You can use the Biopython module when writing the script. Save the script as “Your_index_multi_fasta.py”.



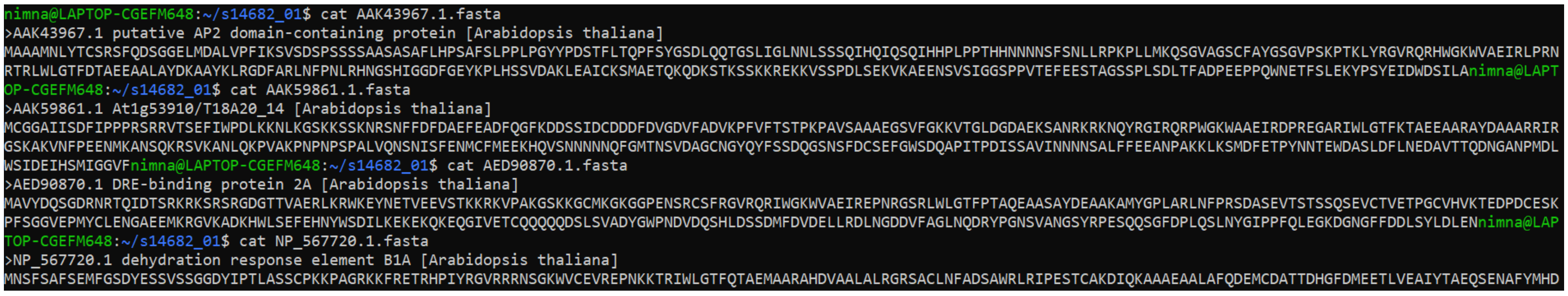
```
nimna@LAPTOP-CGEFM648:~$ cd s14682_01
nimna@LAPTOP-CGEFM648:~/s14682_01$ ls -1
14682_multi_fasta.py
```



```
  GNU nano 6.2                                                                          14682_multi_fasta.py
'''
Retrieve the GenBank records for the "AAK43967.1", "AED90870.1", "NP_567720.1", "AAK59861.1" accessions
Input: "AAK43967.1", "AED90870.1", "NP_567720.1", "AAK59861.1" accessions
Output: The FASTA files named by their respective accession number with the fasta sequence written inside
21/02/2023
Nimna Gamage
Lab 13-Question1_Sub-question1
'''

# Import Biopython sub-modules
from Bio import Entrez
from Bio import SeqIO

# Create a list of accession numbers
acc_list = ["AAK43967.1", "AED90870.1", "NP_567720.1", "AAK59861.1"]

# loop through each accession number in the list
for acc_no in acc_list:
    # provide E-mail
    Entrez.email = "nimnagamage65@gmail.com"
    # retrieve full records from entrez
    handle = Entrez.efetch(db="protein", id=acc_no, rettype="fasta", retmode="text")
    record = SeqIO.read(handle, "fasta")
    # close the handle
    handle.close()
    # open the fasta file named with the respective accession number
    file = open("{}.fasta".format(acc_no), 'w')
    # write the respective fasta sequence in each file
    file.write(">{}\n{}".format(record.description, record.seq))
    # close the file
    file.close()
```

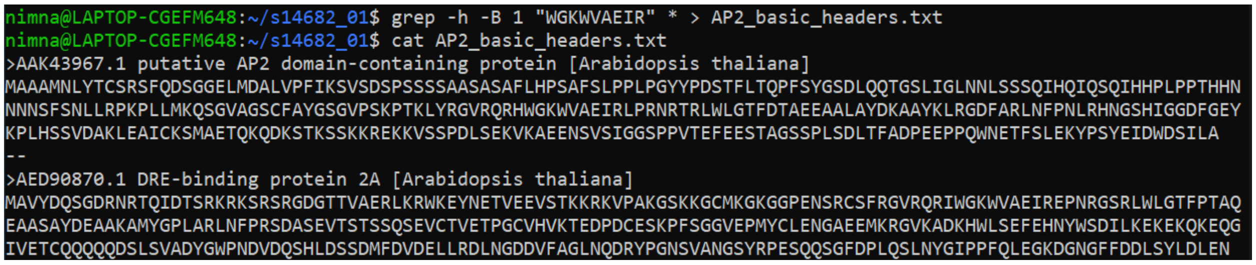


```
nimna@LAPTOP-CGEFM648:~/s14682_01$ python.exe 14682_multi_fasta.py
nimna@LAPTOP-CGEFM648:~/s14682_01$ ls -1
14682_multi_fasta.py
AAK43967.1.fasta
AAK59861.1.fasta
AED90870.1.fasta
NP_567720.1.fasta
```

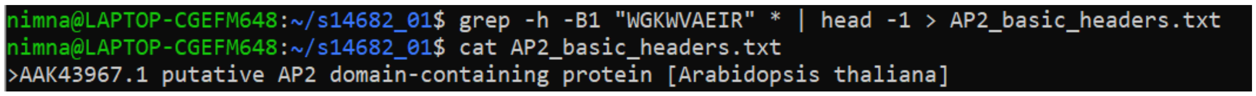


```
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat AAK43967.1.fasta
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
MAAAMNLYTCSRSFQDSGGELMDALVPFIKSVSDSPSSSSAASASAFLHPSAFSLPPLPGYYPDSTFLTQPFSYGSDLQQTGSLIGLNNLSSSQIHQIQSQIHHPLPPTHHNNNNSFSNLLRPKPLLMKQSGVAGSCFAYGSGVPSKPTKLYRGVRQRHWGKWVAEIRLPRN
RTRLWLGTFDTAEEAALAYDKAAYKLRGDFARLNFPNLRHNGSHIGGDFGEYKPLHSSVDAKLEAICKSMAETQKQDKSTKSSKKREKKVSSPDLSEKVKAEENSVSIGGSPPVTEFEESTAGSSPLSDLTFADPEEPPQWNETFSLEKYPSYEIDWDSILAnimna@LAPT
OP-CGEFM648:~/s14682_01$ cat AAK59861.1.fasta
>AAK59861.1 At1g53910/T18A20_14 [Arabidopsis thaliana]
MCGGAIISDFIPPPRSRRVTSEFIWPDLKKNLKGSKKSSKNRSNFFDFDAEFEADFQGFKDDSSIDCDDDFDVGDVFADVKPFVFTSTPKPAVSAAAEGSVFGKKVTGLDGDAEKSANRKRKNQYRGIRQRPWGKWAAEIRDPREGARIWLGTFKTAEEAARAYDAAARRIR
GSKAKVNFPEENMKANSQKRSVKANLQKPVAKPNPNPSPALVQNSNISFENMCFMEEKHQVSNNNNNQFGMTNSVDAGCNGYQYFSSDQGSNSFDCSEFGWSDQAPITPDISSAVINNNNSALFFEEANPAKKLKSMDFETPYNNTEWDASLDFLNEDAVTTQDNGANPMDL
WSIDEIHSMIGGVFnimna@LAPTOP-CGEFM648:~/s14682_01$ cat AED90870.1.fasta
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
MAVYDQSGDRNRTQIDTSRKRKSRSRGDGTTVAERLKRWKEYNETVEEVSTKKRKVPAKGSKKGCMKGKGGPENSRCSFRGVRQRIWGKWVAEIREPNRGSRLWLGTFPTAQEAASAYDEAAKAMYGPLARLNFPRSDASEVTSTSSQSEVCTVETPGCVHVKTEDPDCESK
PFSGGVEPMYCLENGAEEMKRGVKADKHWLSEFEHNYWSDILKEKEKQKEQGIVETCQQQQQDSLSVADYGWPNDVDQSHLDSSDMFDVDELLRDLNGDDVFAGLNQDRYPGNSVANGSYRPESQQSGFDPLQSLNYGIPPFQLEGKDGNGFFDDLSYLDLENnimna@LAP
TOP-CGEFM648:~/s14682_01$ cat NP_567720.1.fasta
>NP_567720.1 dehydration response element B1A [Arabidopsis thaliana]
MNSFSAFSEMFGSDYESSVSSGGDYIPTLASSCPKKPAGRKKFRETRHPIYRGVRRRNSGKWVCEVREPNKKTRIWLGTFQTAEMAARAHDVAALALRGRSACLNFADSAWRLRIPESTCAKDIQKAAAEAALAFQDEMCDATTDHGFDMEETLVEAIYTAEQSENAFYMHD
```

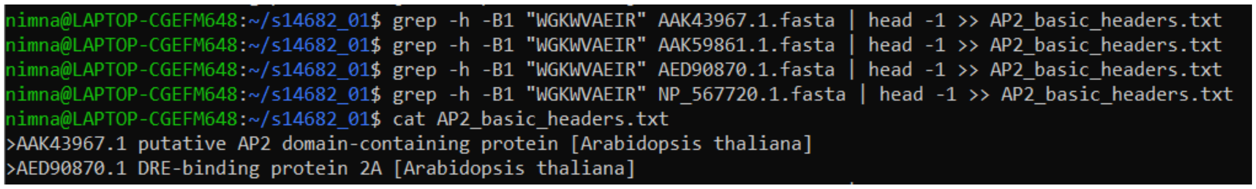
II. Use the grep command to search for the “WGKWVAEIR” amino acid sequence fragment from the AP2/ERF domain in above retrieved sequences and output the FASTA headers of the sequences which contain the domain fragment in a separate file called “AP2_basic_headers.txt”. Write the headers in the space below.



```
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B 1 "WGKWVAEIR" * > AP2_basic_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat AP2_basic_headers.txt
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
MAAAMNLYTCSRSFQDSGGELMDALVPFIKSVSDSPSSSSAASASAFLHPSAFSLPPLPGYYPDSTFLTQPFSYGSDLQQTGSLIGLNNLSSSQIHQIQSQIHHPLPPTHHN
NNNSFSNLLRPKPLLMKQSGVAGSCFAYGSGVPSKPTKLYRGVRQRHWGKWVAEIRLPRNRTRLWLGTFDTAEEAALAYDKAAYKLRGDFARLNFPNLRHNGSHIGGDFGEY
KPLHSSVDAKLEAICKSMAETQKQDKSTKSSKKREKKVSSPDLSEKVKAEENSVSIGGSPPVTEFEESTAGSSPLSDLTFADPEEPPQWNETFSLEKYPSYEIDWDSILA
--
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
MAVYDQSGDRNRTQIDTSRKRKSRSRGDGTTVAERLKRWKEYNETVEEVSTKKRKVPAKGSKKGCMKGKGGPENSRCSFRGVRQRIWGKWVAEIREPNRGSRLWLGTFPTAQ
EAASAYDEAAKAMYGPLARLNFPRSDASEVTSTSSQSEVCTVETPGCVHVKTEDPDCESKPFSGGVEPMYCLENGAEEMKRGVKADKHWLSEFEHNYWSDILKEKEKQKEQG
IVETCQQQQQDSLSVADYGWPNDVDQSHLDSSDMFDVDELLRDLNGDDVFAGLNQDRYPGNSVANGSYRPESQQSGFDPLQSLNYGIPPFQLEGKDGNGFFDDLSYLDLEN
```



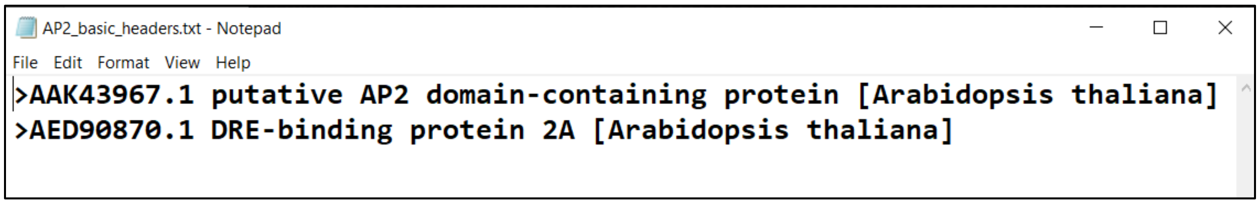
```
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKWVAEIR" * | head -1 > AP2_basic_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat AP2_basic_headers.txt
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
```



```
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKWVAEIR" AAK43967.1.fasta | head -1 >> AP2_basic_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKWVAEIR" AAK59861.1.fasta | head -1 >> AP2_basic_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKWVAEIR" AED90870.1.fasta | head -1 >> AP2_basic_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKWVAEIR" NP_567720.1.fasta | head -1 >> AP2_basic_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat AP2_basic_headers.txt
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
```

AP2_basic_headers.txt file

grep -l "WGKWVAEIR" *.fasta > test.txt



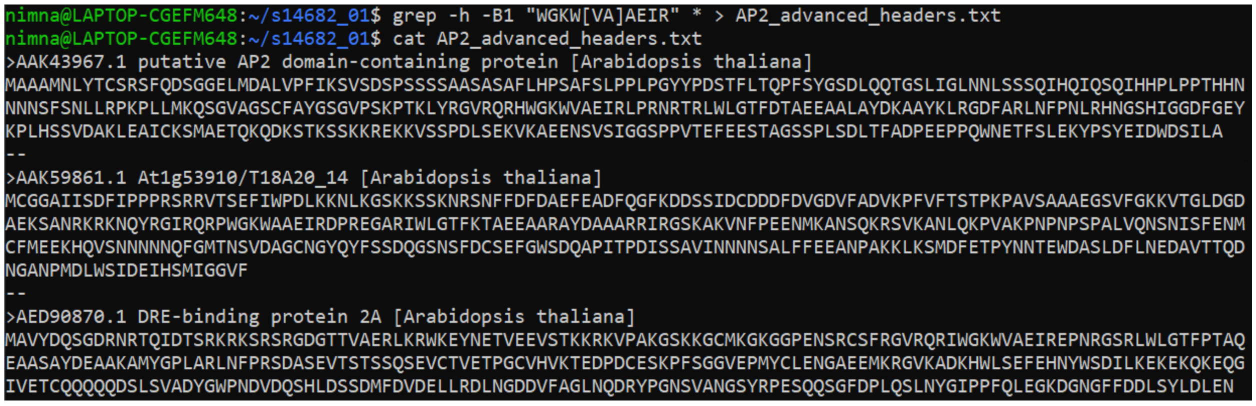
```
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
```

Headers:

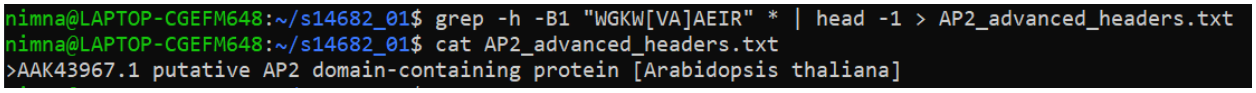
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]

>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]

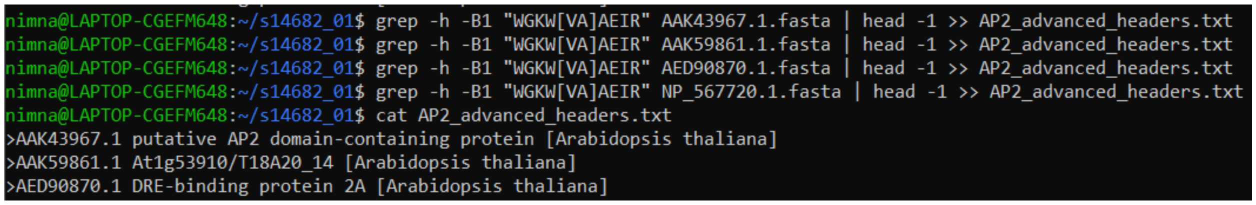
III. Now, modify the above search term to include a REGEX expression to search for “WGKWV/AAEIR” amino acid fragment in the sequences and output the FASTA headers of the sequences which contain the domain fragment in a separate file called “AP2_advanced_headers.txt”. Write the headers in the space below.



```
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKW[VA]AEIR" * > AP2_advanced_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat AP2_advanced_headers.txt
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
MAAAMNLYTCSRSFQDSGGELMDALVPFIKSVSDSPSSSSAASASAFLHPSAFSLPPLPGYYPDSTFLTQPFSYGSDLQQTGSLIGLNNLSSSQIHQIQSQIHHPLPPTHHN
NNNSFSNLLRPKPLLMKQSGVAGSCFAYGSGVPSKPTKLYRGVRQRHWGKWVAEIRLPRNRTRLWLGTFDTAEEAALAYDKAAYKLRGDFARLNFPNLRHNGSHIGGDFGEY
KPLHSSVDAKLEAICKSMAETQKQDKSTKSSKKREKKVSSPDLSEKVKAEENSVSIGGSPPVTEFEESTAGSSPLSDLTFADPEEPPQWNETFSLEKYPSYEIDWDSILA
--
>AAK59861.1 At1g53910/T18A20_14 [Arabidopsis thaliana]
MCGGAIISDFIPPPRSRRVTSEFIWPDLKKNLKGSKKSSKNRSNFFDFDAEFEADFQGFKDDSSIDCDDDFDVGDVFADVKPFVFTSTPKPAVSAAAEGSVFGKKVTGLDGD
AEKSANRKRKNQYRGIRQRPWGKWAAEIRDPREGARIWLGTFKTAEEAARAYDAAARRIRGSKAKVNFPEENMKANSQKRSVKANLQKPVAKPNPNPSPALVQNSNISFENM
CFMEEKHQVSNNNNNQFGMTNSVDAGCNGYQYFSSDQGSNSFDCSEFGWSDQAPITPDISSAVINNNNSALFFEEANPAKKLKSMDFETPYNNTEWDASLDFLNEDAVTTQD
NGANPMDLWSIDEIHSMIGGVF
--
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
MAVYDQSGDRNRTQIDTSRKRKSRSRGDGTTVAERLKRWKEYNETVEEVSTKKRKVPAKGSKKGCMKGKGGPENSRCSFRGVRQRIWGKWVAEIREPNRGSRLWLGTFPTAQ
EAASAYDEAAKAMYGPLARLNFPRSDASEVTSTSSQSEVCTVETPGCVHVKTEDPDCESKPFSGGVEPMYCLENGAEEMKRGVKADKHWLSEFEHNYWSDILKEKEKQKEQG
IVETCQQQQQDSLSVADYGWPNDVDQSHLDSSDMFDVDELLRDLNGDDVFAGLNQDRYPGNSVANGSYRPESQQSGFDPLQSLNYGIPPFQLEGKDGNGFFDDLSYLDLEN
```

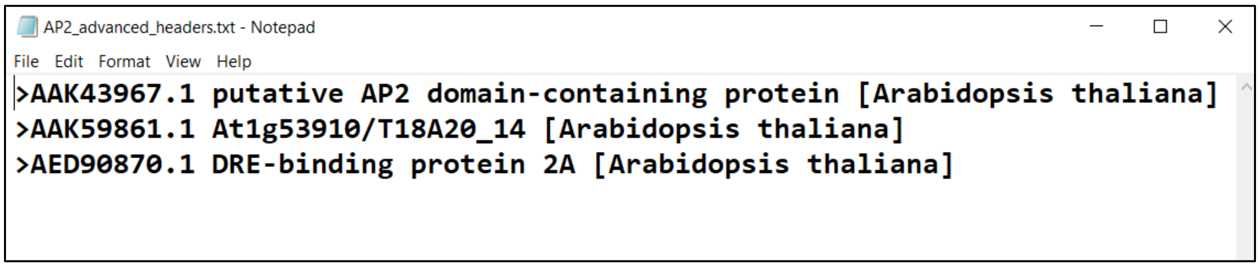


```
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKW[VA]AEIR" * | head -1 > AP2_advanced_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat AP2_advanced_headers.txt
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
```



```
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKW[VA]AEIR" AAK43967.1.fasta | head -1 >> AP2_advanced_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKW[VA]AEIR" AAK59861.1.fasta | head -1 >> AP2_advanced_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKW[VA]AEIR" AED90870.1.fasta | head -1 >> AP2_advanced_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ grep -h -B1 "WGKW[VA]AEIR" NP_567720.1.fasta | head -1 >> AP2_advanced_headers.txt
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat AP2_advanced_headers.txt
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
>AAK59861.1 At1g53910/T18A20_14 [Arabidopsis thaliana]
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
```

AP2_advanced_headers.txt file grep -l "WGKW[VA]AEIR" *.fasta > test.txt



AP2_advanced_headers.txt - Notepad
File Edit Format View Help

```
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
>AAK59861.1 At1g53910/T18A20_14 [Arabidopsis thaliana]
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
```

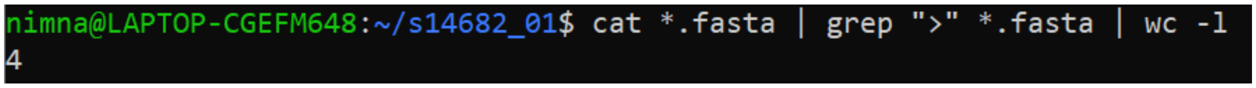
Headers:

>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]

>AAK59861.1 At1g53910/T18A20_14 [Arabidopsis thaliana]

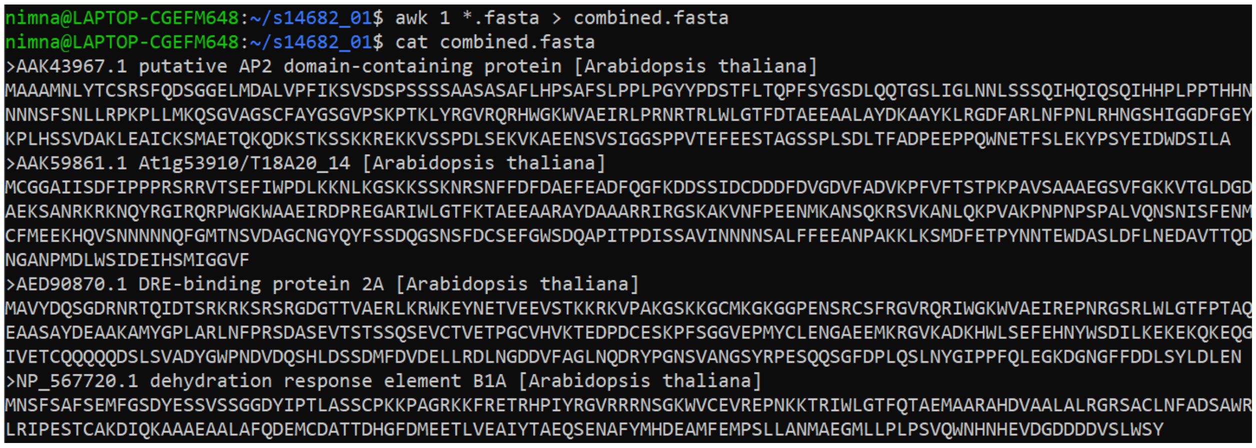
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]

IV. Write a shell command to concatenate the FASTA files downloaded in question (I) and count the number of FASTA files in the concatenated output. Write the count below. Hint: Use the cat, grep, pipe and wc commands appropriately.



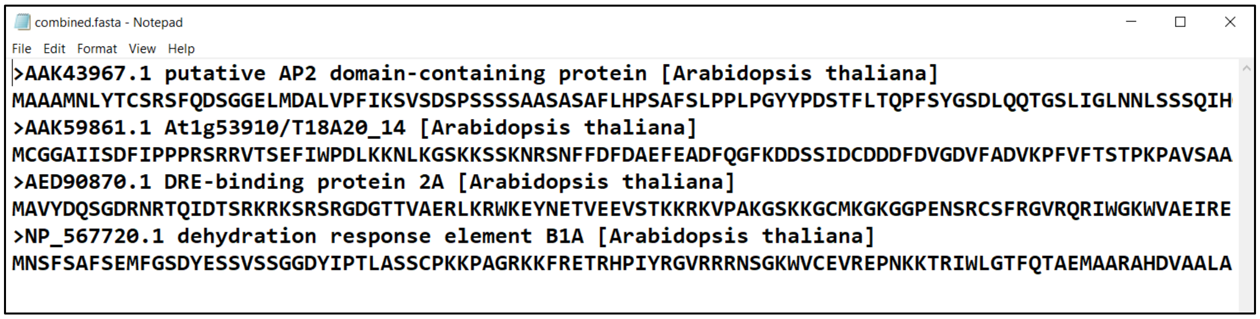
```
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat *.fasta | grep ">" *.fasta | wc -l
4
```

V. Write a shell command to concatenate the FASTA files downloaded in question (I) into a single FASTA file called “combined.fasta”.



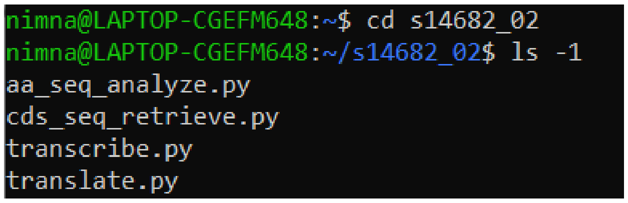
```
nimna@LAPTOP-CGEFM648:~/s14682_01$ awk 1 *.fasta > combined.fasta
nimna@LAPTOP-CGEFM648:~/s14682_01$ cat combined.fasta
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
MAAAMNLYTCSRSFQDSGGELMDALVPFIKSVSDSPSSSSAASASAFLHPSAFSLPPLPGYYPDSTFLTQPFSYGSDLQQTGSLIGLNNLSSSQIHQIQSQIHHPLPPTHHN
NNNSFSNLLRPKPLLMKQSGVAGSCFAYGSGVPSKPTKLYRGVRQRHWGKWVAEIRLPRNRTRLWLGTFDTAEEAALAYDKAAYKLRGDFARLNFPNLRHNGSHIGGDFGEY
KPLHSSVDAKLEAICKSMAETQKQDKSTKSSKKREKKVSSPDLSEKVKAEENSVSIGGSPPVTEFEESTAGSSPLSDLTFADPEEPPQWNETFSLEKYPSYEIDWDSILA
>AAK59861.1 At1g53910/T18A20_14 [Arabidopsis thaliana]
MCGGAIISDFIPPPRSRRVTSEFIWPDLKKNLKGSKKSSKNRSNFFDFDAEFEADFQGFKDDSSIDCDDDFDVGDVFADVKPFVFTSTPKPAVSAAAEGSVFGKKVTGLDGD
AEKSANRKRKNQYRGIRQRPWGKWAAEIRDPREGARIWLGTFKTAEEAARAYDAAARRIRGSKAKVNFPEENMKANSQKRSVKANLQKPVAKPNPNPSPALVQNSNISFENM
CFMEEKHQVSNNNNNQFGMTNSVDAGCNGYQYFSSDQGSNSFDCSEFGWSDQAPITPDISSAVINNNNSALFFEEANPAKKLKSMDFETPYNNTEWDASLDFLNEDAVTTQD
NGANPMDLWSIDEIHSMIGGVF
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
MAVYDQSGDRNRTQIDTSRKRKSRSRGDGTTVAERLKRWKEYNETVEEVSTKKRKVPAKGSKKGCMKGKGGPENSRCSFRGVRQRIWGKWVAEIREPNRGSRLWLGTFPTAQ
EAASAYDEAAKAMYGPLARLNFPRSDASEVTSTSSQSEVCTVETPGCVHVKTEDPDCESKPFSGGVEPMYCLENGAEEMKRGVKADKHWLSEFEHNYWSDILKEKEKQKEQG
IVETCQQQQQDSLSVADYGWPNDVDQSHLDSSDMFDVDELLRDLNGDDVFAGLNQDRYPGNSVANGSYRPESQQSGFDPLQSLNYGIPPFQLEGKDGNGFFDDLSYLDLEN
>NP_567720.1 dehydration response element B1A [Arabidopsis thaliana]
MNSFSAFSEMFGSDYESSVSSGGDYIPTLASSCPKKPAGRKKFRETRHPIYRGVRRRNSGKWVCEVREPNKKTRIWLGTFQTAEMAARAHDVAALALRGRSACLNFADSAWR
LRIPESTCAKDIQKAAAEAALAFQDEMCDATTDHGFDMEETLVEAIYTAEQSENAFYMHDEAMFEMPSLLANMAEGMLLPLPSVQWNHNHEVDGDDDDVSLWSY
```

combined.fasta file



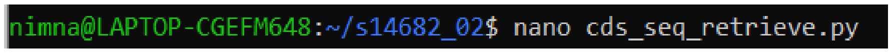
```
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
MAAAMNLYTCSRSFQDSGGELMDALVPFIKSVSDSPSSSSAASASAFLHPSAFSLPPLPGYYPDSTFLTQPFSYGSDLQQTGSLIGLNNLSSSQIH
>AAK59861.1 At1g53910/T18A20_14 [Arabidopsis thaliana]
MCGGAIISDFIPPPRSRRVTSEFIWPDLKKNLKGSKKSSKNRSNFFDFDAEFEADFQGFKDDSSIDCDDDFDVGDVFADVKPFVFTSTPKPAVSAA
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]
MAVYDQSGDRNRTQIDTSRKRKSRSRGDGTTVAERLKRWKEYNETVEEVSTKKRKVPAKGSKKGCMKGKGGPENSRCSFRGVRQRIWGKWVAEIRE
>NP_567720.1 dehydration response element B1A [Arabidopsis thaliana]
MNSFSAFSEMFGSDYESSVSSGGDYIPTLASSCPKKPAGRKKFRETRHPIYRGVRRRNSGKWVCEVREPNKKTRIWLGTFQTAEMAARAHDVAALA
```

2) Write the following Python scripts and assemble them into a bioinformatics pipeline using a shell script. You can use Biopython package when writing the Python scripts. Because you are writing a bioinformatics pipeline, Python scripts must be properly commented and an introduction should be given for each script.

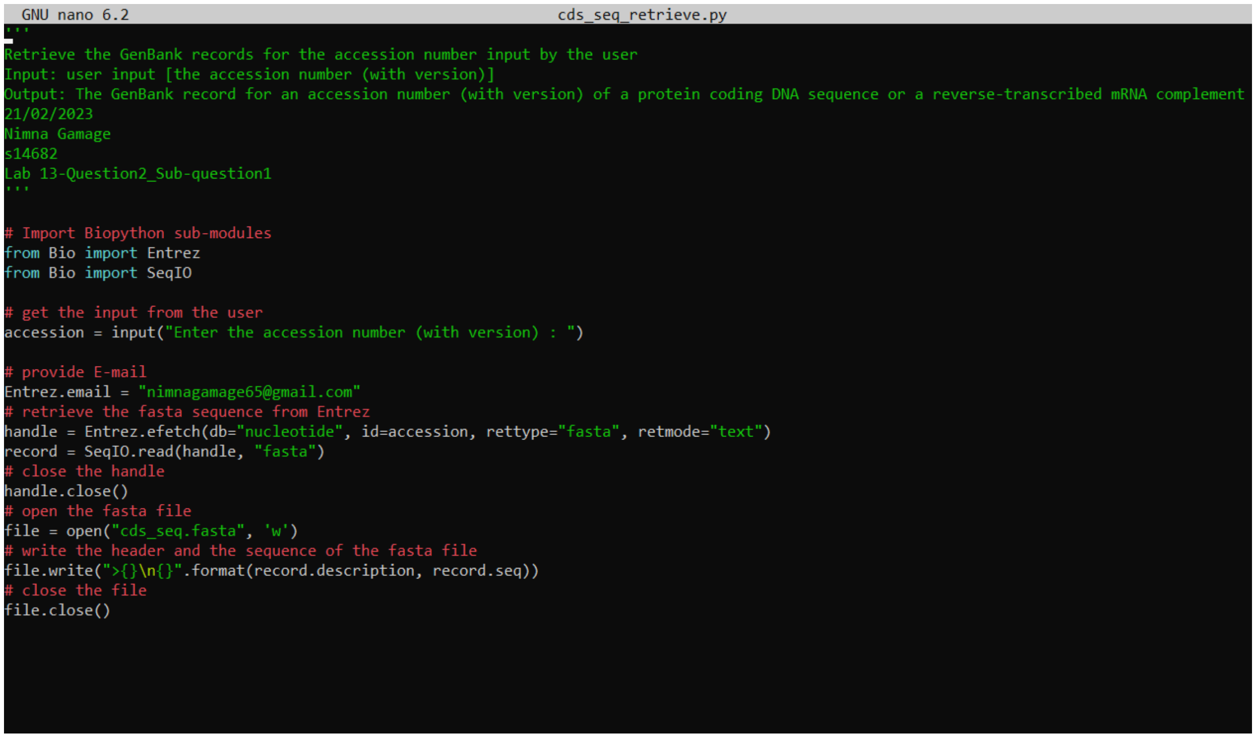


```
nimna@LAPTOP-CGEFM648:~$ cd s14682_02
nimna@LAPTOP-CGEFM648:~/s14682_02$ ls -1
aa_seq_analyze.py
cds_seq_retrieve.py
transcribe.py
translate.py
```

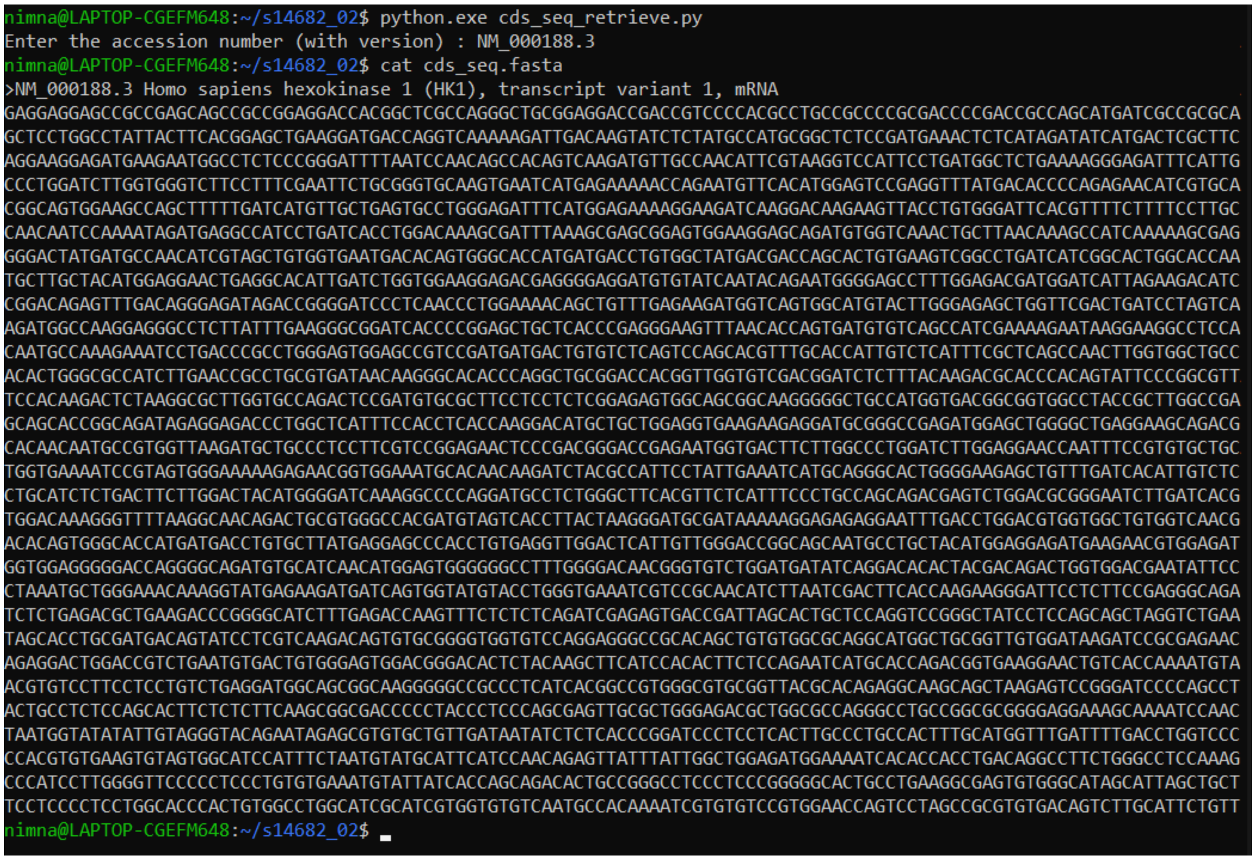
I. First, write a Python script (cds_seq_retrieve.py) to retrieve the GenBank record for an accession number (with version) of a protein coding DNA sequence or a reverse-transcribed mRNA complement given by the user and save the sequence in FASTA format (cds_seq.fasta). The script must prompt the user to input the accession number (with version).



```
nimna@LAPTOP-CGEFM648:~/s14682_02$ nano cds_seq_retrieve.py
```

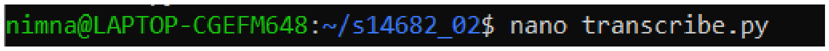


```
GNU nano 6.2                                              cds_seq_retrieve.py
'''
Retrieve the GenBank records for the accession number input by the user
Input: user input [the accession number (with version)]
Output: The GenBank record for an accession number (with version) of a protein coding DNA sequence or a reverse-transcribed mRNA complement
21/02/2023
Nimna Gamage
s14682
Lab 13-Question2_Sub-question1
'''

# Import Biopython sub-modules
from Bio import Entrez
from Bio import SeqIO

# get the input from the user
accession = input("Enter the accession number (with version) : ")

# provide E-mail
Entrez.email = "nimnagamage65@gmail.com"
# retrieve the fasta sequence from Entrez
handle = Entrez.efetch(db="nucleotide", id=accession, rettype="fasta", retmode="text")
record = SeqIO.read(handle, "fasta")
# close the handle
handle.close()
# open the fasta file
file = open("cds_seq.fasta", 'w')
# write the header and the sequence of the fasta file
file.write(">{}\n{}".format(record.description, record.seq))
# close the file
file.close()
```

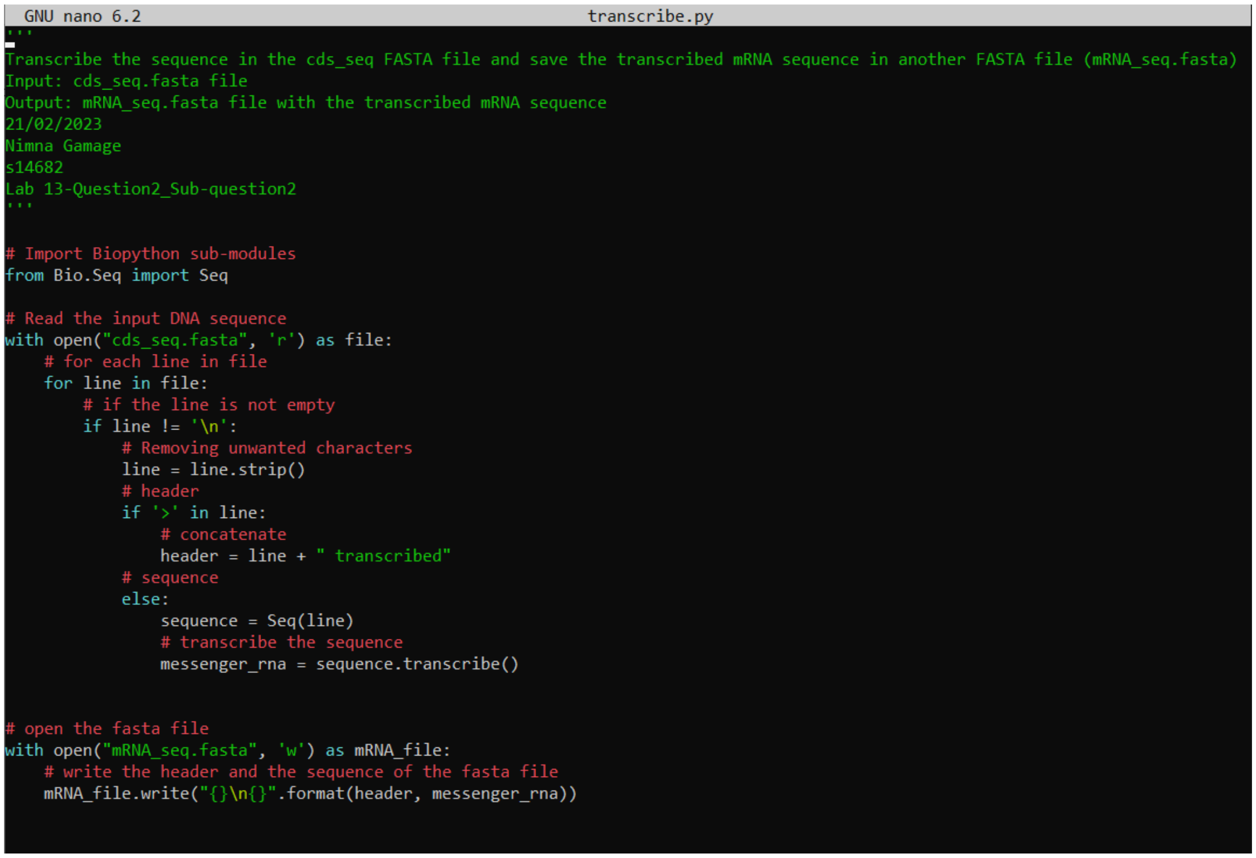


```
nimna@LAPTOP-CGEFM648:~/s14682_02$ python.exe cds_seq_retrieve.py
Enter the accession number (with version) : NM_000188.3
nimna@LAPTOP-CGEFM648:~/s14682_02$ cat cds_seq.fasta
>NM_000188.3 Homo sapiens hexokinase 1 (HK1), transcript variant 1, mRNA
GAGGAGGAGCCGCCGAGCAGCCGCCGGAGGACCACGGCTCGCCAGGGCTGCGGAGGACCGACCGTCCCCACGCCTGCCGCCCCGCGACCCCGACCGCCAGCATGATCGCCGCGCA
GCTCCTGGCCTATTACTTCACGGAGCTGAAGGATGACCAGGTCAAAAAGATTGACAAGTATCTCTATGCCATGCGGCTCTCCGATGAAACTCTCATAGATATCATGACTCGCTTC
AGGAAGGAGATGAAGAATGGCCTCTCCCGGGATTTTAATCCAACAGCCACAGTCAAGATGTTGCCAACATTCGTAAGGTCCATTCCTGATGGCTCTGAAAAGGGAGATTTCATTG
CCCTGGATCTTGGTGGGTCTTCCTTTCGAATTCTGCGGGTGCAAGTGAATCATGAGAAAAACCAGAATGTTCACATGGAGTCCGAGGTTTATGACACCCCAGAGAACATCGTGCA
CGGCAGTGGAAGCCAGCTTTTTGATCATGTTGCTGAGTGCCTGGGAGATTTCATGGAGAAAAGGAAGATCAAGGACAAGAAGTTACCTGTGGGATTCACGTTTTCTTTTCCTTGC
CAACAATCCAAAATAGATGAGGCCATCCTGATCACCTGGACAAAGCGATTTAAAGCGAGCGGAGTGGAAGGAGCAGATGTGGTCAAACTGCTTAACAAAGCCATCAAAAAGCGAG
GGGACTATGATGCCAACATCGTAGCTGTGGTGAATGACACAGTGGGCACCATGATGACCTGTGGCTATGACGACCAGCACTGTGAAGTCGGCCTGATCATCGGCACTGGCACCAA
TGCTTGCTACATGGAGGAACTGAGGCACATTGATCTGGTGGAAGGAGACGAGGGGAGGATGTGTATCAATACAGAATGGGGAGCCTTTGGAGACGATGGATCATTAGAAGACATC
CGGACAGAGTTTGACAGGGAGATAGACCGGGGATCCCTCAACCCTGGAAAACAGCTGTTTGAGAAGATGGTCAGTGGCATGTACTTGGGAGAGCTGGTTCGACTGATCCTAGTCA
AGATGGCCAAGGAGGGCCTCTTATTTGAAGGGCGGATCACCCCGGAGCTGCTCACCCGAGGGAAGTTTAACACCAGTGATGTGTCAGCCATCGAAAAGAATAAGGAAGGCCTCCA
CAATGCCAAAGAAATCCTGACCCGCCTGGGAGTGGAGCCGTCCGATGATGACTGTGTCTCAGTCCAGCACGTTTGCACCATTGTCTCATTTCGCTCAGCCAACTTGGTGGCTGCC
ACACTGGGCGCCATCTTGAACCGCCTGCGTGATAACAAGGGCACACCCAGGCTGCGGACCACGGTTGGTGTCGACGGATCTCTTTACAAGACGCACCCACAGTATTCCCGGCGTT
TCCACAAGACTCTAAGGCGCTTGGTGCCAGACTCCGATGTGCGCTTCCTCCTCTCGGAGAGTGGCAGCGGCAAGGGGGCTGCCATGGTGACGGCGGTGGCCTACCGCTTGGCCGA
GCAGCACCGGCAGATAGAGGAGACCCTGGCTCATTTCCACCTCACCAAGGACATGCTGCTGGAGGTGAAGAAGAGGATGCGGGCCGAGATGGAGCTGGGGCTGAGGAAGCAGACG
CACAACAATGCCGTGGTTAAGATGCTGCCCTCCTTCGTCCGGAGAACTCCCGACGGGACCGAGAATGGTGACTTCTTGGCCCTGGATCTTGGAGGAACCAATTTCCGTGTGCTGC
TGGTGAAAATCCGTAGTGGGAAAAAGAGAACGGTGGAAATGCACAACAAGATCTACGCCATTCCTATTGAAATCATGCAGGGCACTGGGGAAGAGCTGTTTGATCACATTGTCTC
CTGCATCTCTGACTTCTTGGACTACATGGGGATCAAAGGCCCCAGGATGCCTCTGGGCTTCACGTTCTCATTTCCCTGCCAGCAGACGAGTCTGGACGCGGGAATCTTGATCACG
TGGACAAAGGGTTTTAAGGCAACAGACTGCGTGGGCCACGATGTAGTCACCTTACTAAGGGATGCGATAAAAAGGAGAGAGGAATTTGACCTGGACGTGGTGGCTGTGGTCAACG
ACACAGTGGGCACCATGATGACCTGTGCTTATGAGGAGCCCACCTGTGAGGTTGGACTCATTGTTGGGACCGGCAGCAATGCCTGCTACATGGAGGAGATGAAGAACGTGGAGAT
GGTGGAGGGGGACCAGGGGCAGATGTGCATCAACATGGAGTGGGGGGCCTTTGGGGACAACGGGTGTCTGGATGATATCAGGACACACTACGACAGACTGGTGGACGAATATTCC
CTAAATGCTGGGAAACAAAGGTATGAGAAGATGATCAGTGGTATGTACCTGGGTGAAATCGTCCGCAACATCTTAATCGACTTCACCAAGAAGGGATTCCTCTTCCGAGGGCAGA
TCTCTGAGACGCTGAAGACCCGGGGCATCTTTGAGACCAAGTTTCTCTCTCAGATCGAGAGTGACCGATTAGCACTGCTCCAGGTCCGGGCTATCCTCCAGCAGCTAGGTCTGAA
TAGCACCTGCGATGACAGTATCCTCGTCAAGACAGTGTGCGGGGTGGTGTCCAGGAGGGCCGCACAGCTGTGTGGCGCAGGCATGGCTGCGGTTGTGGATAAGATCCGCGAGAAC
AGAGGACTGGACCGTCTGAATGTGACTGTGGGAGTGGACGGGACACTCTACAAGCTTCATCCACACTTCTCCAGAATCATGCACCAGACGGTGAAGGAACTGTCACCAAAATGTA
ACGTGTCCTTCCTCCTGTCTGAGGATGGCAGCGGCAAGGGGGCCGCCCTCATCACGGCCGTGGGCGTGCGGTTACGCACAGAGGCAAGCAGCTAAGAGTCCGGGATCCCCAGCCT
ACTGCCTCTCCAGCACTTCTCTCTTCAAGCGGCGACCCCCTACCCTCCCAGCGAGTTGCGCTGGGAGACGCTGGCGCCAGGGCCTGCCGGCGCGGGGAGGAAAGCAAAATCCAAC
TAATGGTATATATTGTAGGGTACAGAATAGAGCGTGTGCTGTTGATAATATCTCTCACCCGGATCCCTCCTCACTTGCCCTGCCACTTTGCATGGTTTGATTTTGACCTGGTCCC
CCACGTGTGAAGTGTAGTGGCATCCATTTCTAATGTATGCATTCATCCAACAGAGTTATTTATTGGCTGGAGATGGAAAATCACACCACCTGACAGGCCTTCTGGGCCTCCAAAG
CCCATCCTTGGGGTTCCCCCTCCCTGTGTGAAATGTATTATCACCAGCAGACACTGCCGGGCCTCCCTCCCGGGGGCACTGCCTGAAGGCGAGTGTGGGCATAGCATTAGCTGCT
TCCTCCCCTCCTGGCACCCACTGTGGCCTGGCATCGCATCGTGGTGTGTCAATGCCACAAAATCGTGTGTCCGTGGAACCAGTCCTAGCCGCGTGTGACAGTCTTGCATTCTGTT
nimna@LAPTOP-CGEFM648:~/s14682_02$
```

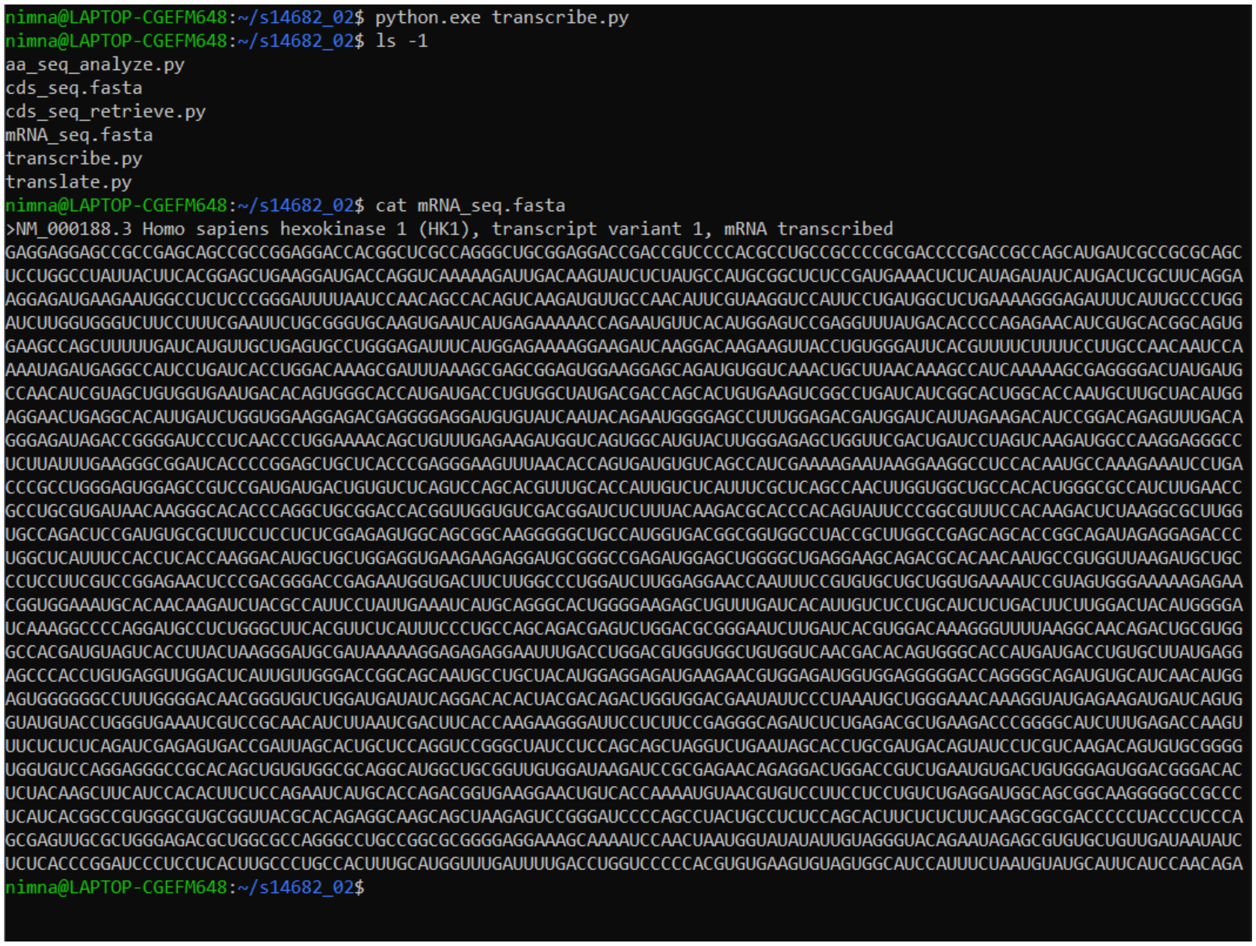
II. Then, write a Python script (transcribe.py) to transcribe the above sequence in the FASTA file and save the transcribed mRNA sequence in another FASTA file (mRNA_seq.fasta). The FASTA header must contain the added word "transcribed" at the end. The program should read the "cds_seq.fasta" file.



```
nimna@LAPTOP-CGEFM648:~/s14682_02$ nano transcribe.py
```

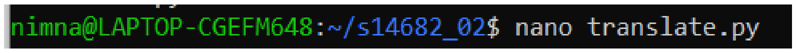


```
  GNU nano 6.2                                   transcribe.py
'''
Transcribe the sequence in the cds_seq FASTA file and save the transcribed mRNA sequence in another FASTA file (mRNA_seq.fasta)
Input: cds_seq.fasta file
Output: mRNA_seq.fasta file with the transcribed mRNA sequence
21/02/2023
Nimna Gamage
s14682
Lab 13-Question2_Sub-question2
'''

# Import Biopython sub-modules
from Bio.Seq import Seq

# Read the input DNA sequence
with open("cds_seq.fasta", 'r') as file:
    # for each line in file
    for line in file:
        # if the line is not empty
        if line != '\n':
            # Removing unwanted characters
            line = line.strip()
            # header
            if '>' in line:
                # concatenate
                header = line + " transcribed"
            # sequence
            else:
                sequence = Seq(line)
                # transcribe the sequence
                messenger_rna = sequence.transcribe()


# open the fasta file
with open("mRNA_seq.fasta", 'w') as mRNA_file:
    # write the header and the sequence of the fasta file
    mRNA_file.write("{}\n{}".format(header, messenger_rna))
```

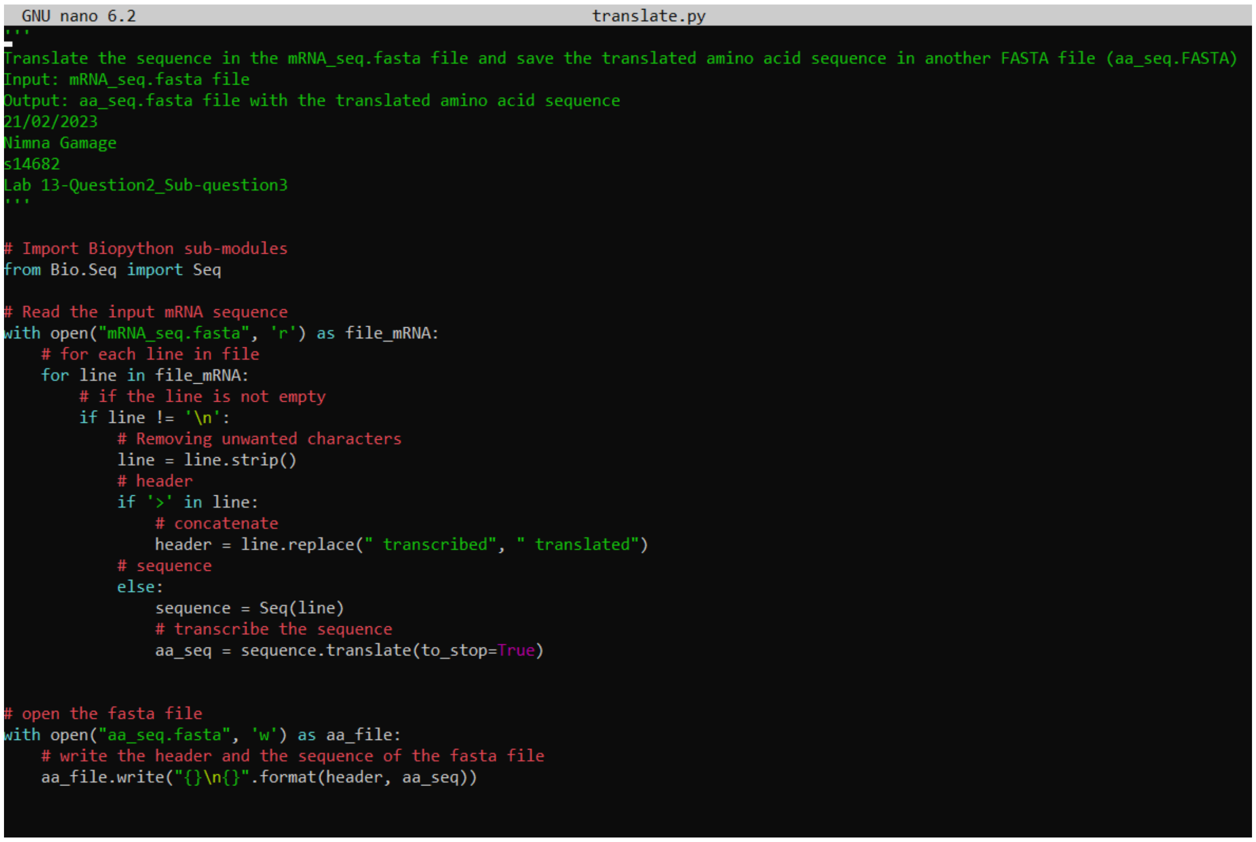


```
nimna@LAPTOP-CGEFM648:~/s14682_02$ python.exe transcribe.py
nimna@LAPTOP-CGEFM648:~/s14682_02$ ls -1
aa_seq_analyze.py
cds_seq.fasta
cds_seq_retrieve.py
mRNA_seq.fasta
transcribe.py
translate.py
nimna@LAPTOP-CGEFM648:~/s14682_02$ cat mRNA_seq.fasta
>NM_000188.3 Homo sapiens hexokinase 1 (HK1), transcript variant 1, mRNA transcribed
GAGGAGGAGCCGCCGAGCAGCCGCCGGAGGACCACGGCUCGCCAGGGCUGCGGAGGACCGACCGUCCCCACGCCUGCCGCCCCGCGACCCCGACCGCCAGCAUGAUCGCCGCGCAGC
UCCUGGCCUAUUACUUCACGGAGCUGAAGGAUGACCAGGUCAAAAAGAUUGACAAGUAUCUCUAUGCCAUGCGGCUCUCCGAUGAAACUCUCAUAGAUAUCAUGACUCGCUUCAGGA
AGGAGAUGAAGAAUGGCCUCUCCCGGGAUUUUAAUCCAACAGCCACAGUCAAGAUGUUGCCAACAUUCGUAAGGUCCAUUCCUGAUGGCUCUGAAAAGGGAGAUUUCAUUGCCCUGG
AUCUUGGUGGGUCUUCCUUUCGAAUUCUGCGGGUGCAAGUGAAUCAUGAGAAAAACCAGAAUGUUCACAUGGAGUCCGAGGUUUAUGACACCCCAGAGAACAUCGUGCACGGCAGUG
GAAGCCAGCUUUUUGAUCAUGUUGCUGAGUGCCUGGGAGAUUUCAUGGAGAAAAGGAAGAUCAAGGACAAGAAGUUACCUGUGGGAUUCACGUUUUCUUUUCCUUGCCAACAAUCCA
AAAUAGAUGAGGCCAUCCUGAUCACCUGGACAAAGCGAUUUAAAGCGAGCGGAGUGGAAGGAGCAGAUGUGGUCAAACUGCUUAACAAAGCCAUCAAAAAGCGAGGGGACUAUGAUG
CCAACAUCGUAGCUGUGGUGAAUGACACAGUGGGCACCAUGAUGACCUGUGGCUAUGACGACCAGCACUGUGAAGUCGGCCUGAUCAUCGGCACUGGCACCAAUGCUUGCUACAUGG
AGGAACUGAGGCACAUUGAUCUGGUGGAAGGAGACGAGGGGAGGAUGUGUAUCAAUACAGAAUGGGGAGCCUUUGGAGACGAUGGAUCAUUAGAAGACAUCCGGACAGAGUUUGACA
GGGAGAUAGACCGGGGAUCCCUCAACCCUGGAAAACAGCUGUUUGAGAAGAUGGUCAGUGGCAUGUACUUGGGAGAGCUGGUUCGACUGAUCCUAGUCAAGAUGGCCAAGGAGGGCC
UCUUAUUUGAAGGGCGGAUCACCCCGGAGCUGCUCACCCGAGGGAAGUUUAACACCAGUGAUGUGUCAGCCAUCGAAAAGAAUAAGGAAGGCCUCCACAAUGCCAAAGAAAUCCUGA
CCCGCCUGGGAGUGGAGCCGUCCGAUGAUGACUGUGUCUCAGUCCAGCACGUUUGCACCAUUGUCUCAUUUCGCUCAGCCAACUUGGUGGCUGCCACACUGGGCGCCAUCUUGAACC
GCCUGCGUGAUAACAAGGGCACACCCAGGCUGCGGACCACGGUUGGUGUCGACGGAUCUCUUUACAAGACGCACCCACAGUAUUCCCGGCGUUUCCACAAGACUCUAAGGCGCUUGG
UGCCAGACUCCGAUGUGCGCUUCCUCCUCUCGGAGAGUGGCAGCGGCAAGGGGGCUGCCAUGGUGACGGCGGUGGCCUACCGCUUGGCCGAGCAGCACCGGCAGAUAGAGGAGACCC
UGGCUCAUUUCCACCUCACCAAGGACAUGCUGCUGGAGGUGAAGAAGAGGAUGCGGGCCGAGAUGGAGCUGGGGCUGAGGAAGCAGACGCACAACAAUGCCGUGGUUAAGAUGCUGC
CCUCCUUCGUCCGGAGAACUCCCGACGGGACCGAGAAUGGUGACUUCUUGGCCCUGGAUCUUGGAGGAACCAAUUUCCGUGUGCUGCUGGUGAAAAUCCGUAGUGGGAAAAAGAGAA
CGGUGGAAAUGCACAACAAGAUCUACGCCAUUCCUAUUGAAAUCAUGCAGGGCACUGGGGAAGAGCUGUUUGAUCACAUUGUCUCCUGCAUCUCUGACUUCUUGGACUACAUGGGGA
UCAAAGGCCCCAGGAUGCCUCUGGGCUUCACGUUCUCAUUUCCCUGCCAGCAGACGAGUCUGGACGCGGGAAUCUUGAUCACGUGGACAAAGGGUUUUAAGGCAACAGACUGCGUGG
GCCACGAUGUAGUCACCUUACUAAGGGAUGCGAUAAAAAGGAGAGAGGAAUUUGACCUGGACGUGGUGGCUGUGGUCAACGACACAGUGGGCACCAUGAUGACCUGUGCUUAUGAGG
AGCCCACCUGUGAGGUUGGACUCAUUGUUGGGACCGGCAGCAAUGCCUGCUACAUGGAGGAGAUGAAGAACGUGGAGAUGGUGGAGGGGGACCAGGGGCAGAUGUGCAUCAACAUGG
AGUGGGGGGCCUUUGGGGACAACGGGUGUCUGGAUGAUAUCAGGACACACUACGACAGACUGGUGGACGAAUAUUCCCUAAAUGCUGGGAAACAAAGGUAUGAGAAGAUGAUCAGUG
GUAUGUACCUGGGUGAAAUCGUCCGCAACAUCUUAAUCGACUUCACCAAGAAGGGAUUCCUCUUCCGAGGGCAGAUCUCUGAGACGCUGAAGACCCGGGGCAUCUUUGAGACCAAGU
UUCUCUCUCAGAUCGAGAGUGACCGAUUAGCACUGCUCCAGGUCCGGGCUAUCCUCCAGCAGCUAGGUCUGAAUAGCACCUGCGAUGACAGUAUCCUCGUCAAGACAGUGUGCGGGG
UGGUGUCCAGGAGGGCCGCACAGCUGUGUGGCGCAGGCAUGGCUGCGGUUGUGGAUAAGAUCCGCGAGAACAGAGGACUGGACCGUCUGAAUGUGACUGUGGGAGUGGACGGGACAC
UCUACAAGCUUCAUCCACACUUCUCCAGAAUCAUGCACCAGACGGUGAAGGAACUGUCACCAAAAUGUAACGUGUCCUUCCUCCUGUCUGAGGAUGGCAGCGGCAAGGGGGCCGCCC
UCAUCACGGCCGUGGGCGUGCGGUUACGCACAGAGGCAAGCAGCUAAGAGUCCGGGAUCCCCAGCCUACUGCCUCUCCAGCACUUCUCUCUUCAAGCGGCGACCCCCUACCCUCCCA
GCGAGUUGCGCUGGGAGACGCUGGCGCCAGGGCCUGCCGGCGCGGGGAGGAAAGCAAAAUCCAACUAAUGGUAUAUAUUGUAGGGUACAGAAUAGAGCGUGUGCUGUUGAUAAUAUC
UCUCACCCGGAUCCCUCCUCACUUGCCCUGCCACUUUGCAUGGUUUGAUUUUGACCUGGUCCCCCACGUGUGAAGUGUAGUGGCAUCCAUUUCUAAUGUAUGCAUUCAUCCAACAGA
nimna@LAPTOP-CGEFM648:~/s14682_02$
```

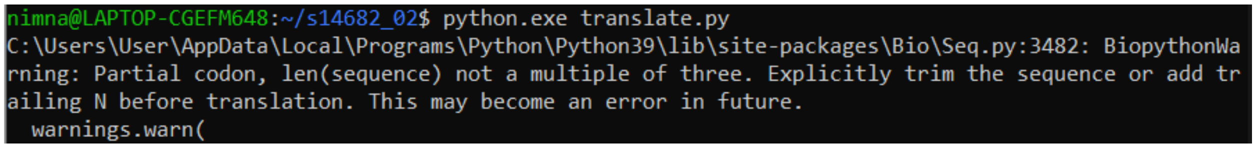
III. Then, write a Python script (translate.py) to translate the above sequence in the FASTA file and save the translated amino acid sequence in another FASTA file (aa_seq.FASTA). The FASTA header must contain the added word "translated" at the end. The script must read the "mRNA_seq.fasta" file.
Hint: for this example, it is not needed to start the amino acid sequence with methionine. Simply translate the mRNA sequence using translate() function in Biopython.



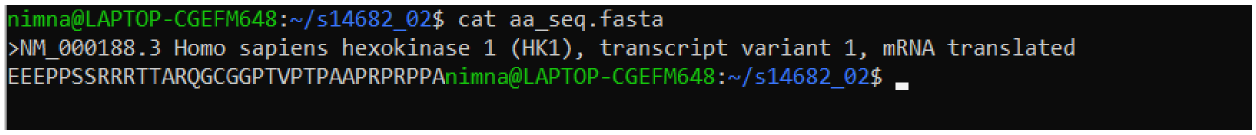
```
nimna@LAPTOP-CGEFM648:~/s14682_02$ nano translate.py
```



```
  GNU nano 6.2                                         translate.py
'''
Translate the sequence in the mRNA_seq.fasta file and save the translated amino acid sequence in another FASTA file (aa_seq.FASTA)
Input: mRNA_seq.fasta file
Output: aa_seq.fasta file with the translated amino acid sequence
21/02/2023
Nimna Gamage
s14682
Lab 13-Question2_Sub-question3
'''

# Import Biopython sub-modules
from Bio.Seq import Seq

# Read the input mRNA sequence
with open("mRNA_seq.fasta", 'r') as file_mRNA:
    # for each line in file
    for line in file_mRNA:
        # if the line is not empty
        if line != '\n':
            # Removing unwanted characters
            line = line.strip()
            # header
            if '>' in line:
                # concatenate
                header = line.replace(" transcribed", " translated")
            # sequence
            else:
                sequence = Seq(line)
                # transcribe the sequence
                aa_seq = sequence.translate(to_stop=True)


# open the fasta file
with open("aa_seq.fasta", 'w') as aa_file:
    # write the header and the sequence of the fasta file
    aa_file.write("{}\n{}".format(header, aa_seq))
```

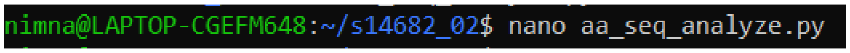


```
nimna@LAPTOP-CGEFM648:~/s14682_02$ python.exe translate.py
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\Bio\Seq.py:3482: BiopythonWa
rning: Partial codon, len(sequence) not a multiple of three. Explicitly trim the sequence or add tr
ailing N before translation. This may become an error in future.
  warnings.warn(
```

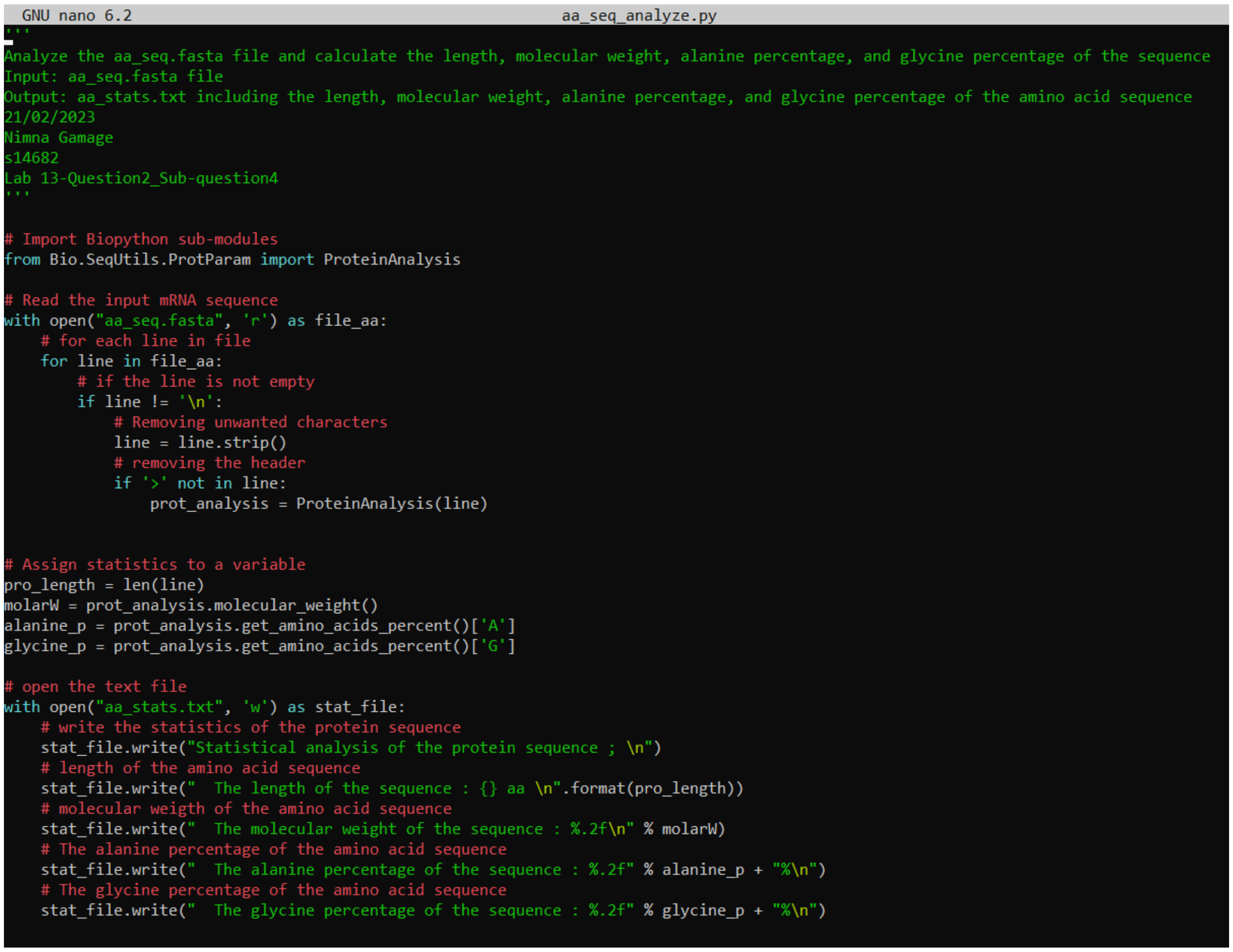


```
nimna@LAPTOP-CGEFM648:~/s14682_02$ cat aa_seq.fasta
>NM_000188.3 Homo sapiens hexokinase 1 (HK1), transcript variant 1, mRNA translated
EEEPPSSRRRTTARQGCGGPTVPTPAAPRPRPPAnimna@LAPTOP-CGEFM648:~/s14682_02$
```

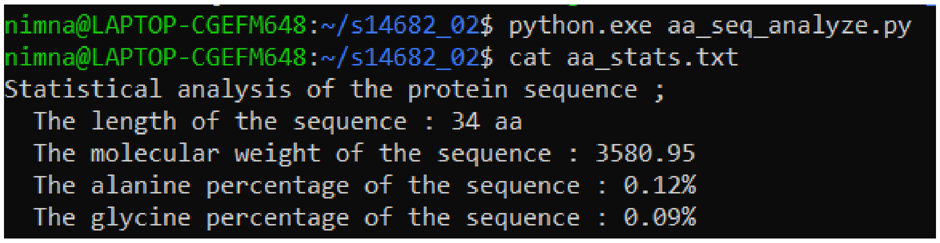
IV. Finally, write a Python script (aa_seq_analyze.py) to analyze the aa_seq.fasta file and calculate the length, molecular weight, alanine percentage, and glycine percentage of the sequence. Save the calculated parameters in a new text file called "aa_stats.txt". The script must read the "aa_seq.fasta" file as the input.
Hint: You can use Biopython for above calculations. Find out the specific sub module for protein sequence analysis.



```
nimna@LAPTOP-CGEFM648:~/s14682_02$ nano aa_seq_analyze.py
```

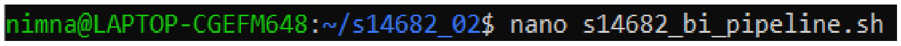


```
  GNU nano 6.2                                                    aa_seq_analyze.py
'''
Analyze the aa_seq.fasta file and calculate the length, molecular weight, alanine percentage, and glycine percentage of the sequence
Input: aa_seq.fasta file
Output: aa_stats.txt including the length, molecular weight, alanine percentage, and glycine percentage of the amino acid sequence
21/02/2023
Nimna Gamage
s14682
Lab 13-Question2_Sub-question4
'''

# Import Biopython sub-modules
from Bio.SeqUtils.ProtParam import ProteinAnalysis

# Read the input mRNA sequence
with open("aa_seq.fasta", 'r') as file_aa:
    # for each line in file
    for line in file_aa:
        # if the line is not empty
        if line != '\n':
            # Removing unwanted characters
            line = line.strip()
            # removing the header
            if '>' not in line:
                prot_analysis = ProteinAnalysis(line)


# Assign statistics to a variable
pro_length = len(line)
molarW = prot_analysis.molecular_weight()
alanine_p = prot_analysis.get_amino_acids_percent()['A']
glycine_p = prot_analysis.get_amino_acids_percent()['G']

# open the text file
with open("aa_stats.txt", 'w') as stat_file:
    # write the statistics of the protein sequence
    stat_file.write("Statistical analysis of the protein sequence ; \n")
    # length of the amino acid sequence
    stat_file.write("  The length of the sequence : {} aa \n".format(pro_length))
    # molecular weigth of the amino acid sequence
    stat_file.write("  The molecular weight of the sequence : %.2f\n" % molarW)
    # The alanine percentage of the amino acid sequence
    stat_file.write("  The alanine percentage of the sequence : %.2f" % alanine_p + "%\n")
    # The glycine percentage of the amino acid sequence
    stat_file.write("  The glycine percentage of the sequence : %.2f" % glycine_p + "%\n")
```



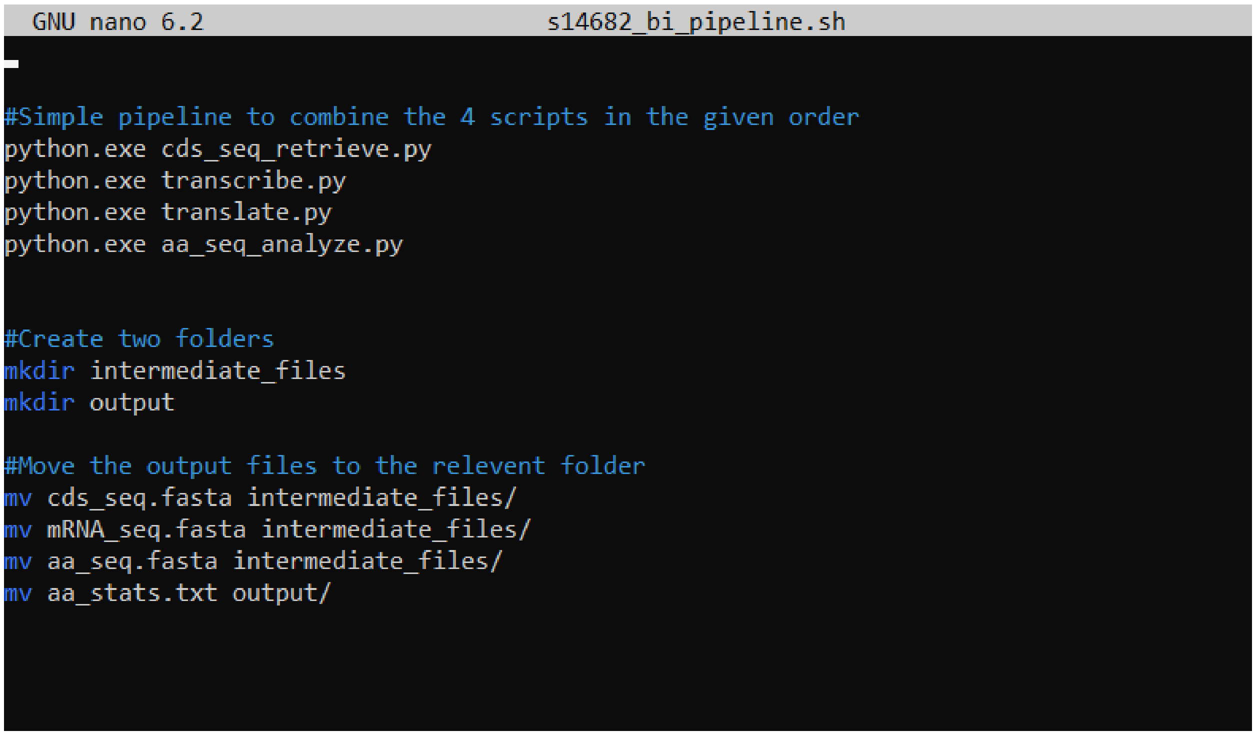
```
nimna@LAPTOP-CGEFM648:~/s14682_02$ python.exe aa_seq_analyze.py
nimna@LAPTOP-CGEFM648:~/s14682_02$ cat aa_stats.txt
Statistical analysis of the protein sequence ;
  The length of the sequence : 34 aa
  The molecular weight of the sequence : 3580.95
  The alanine percentage of the sequence : 0.12%
  The glycine percentage of the sequence : 0.09%
```

V. Now, using a shell script, build a simple pipeline to combine the above 4 scripts in the given order. Further, using the same shell script, create two folders: intermediate_files and output. Move the “cds_seq.fasta”, “mRNA_seq.fasta”, and “aa_seq.fasta ” into the intermediate_files folder and the final output file: “aa_stats.txt” into the output folder. Save the shell script as “your_index_bi_pipeline.sh”. Use the “NM_000188.3” accession as the input to the pipeline, which is for human hexokinase 1 gene. Write the amino acid statistics you calculated below, which would be in the output folder.



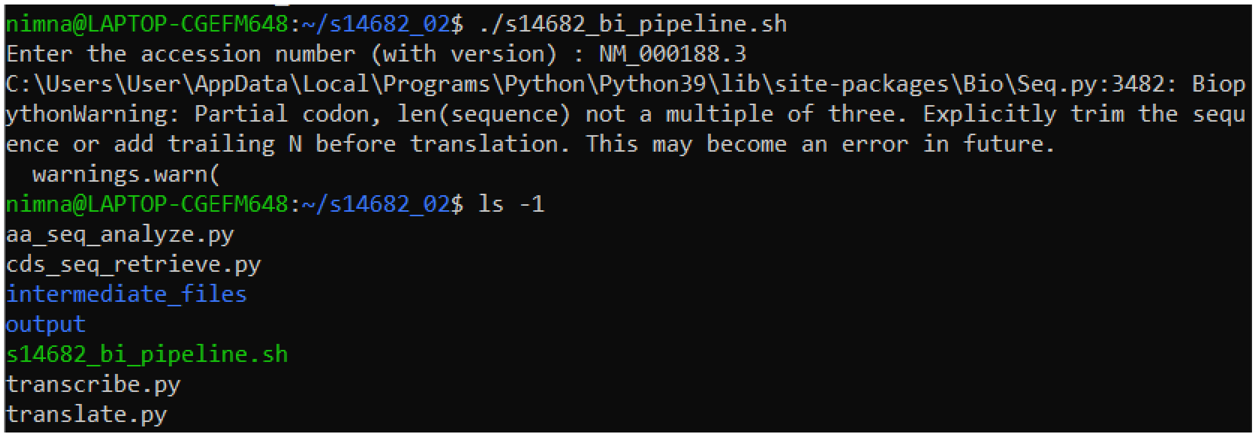
```
nimna@LAPTOP-CGEFM648:~/s14682_02$ nano s14682_bi_pipeline.sh
```

Shell script;



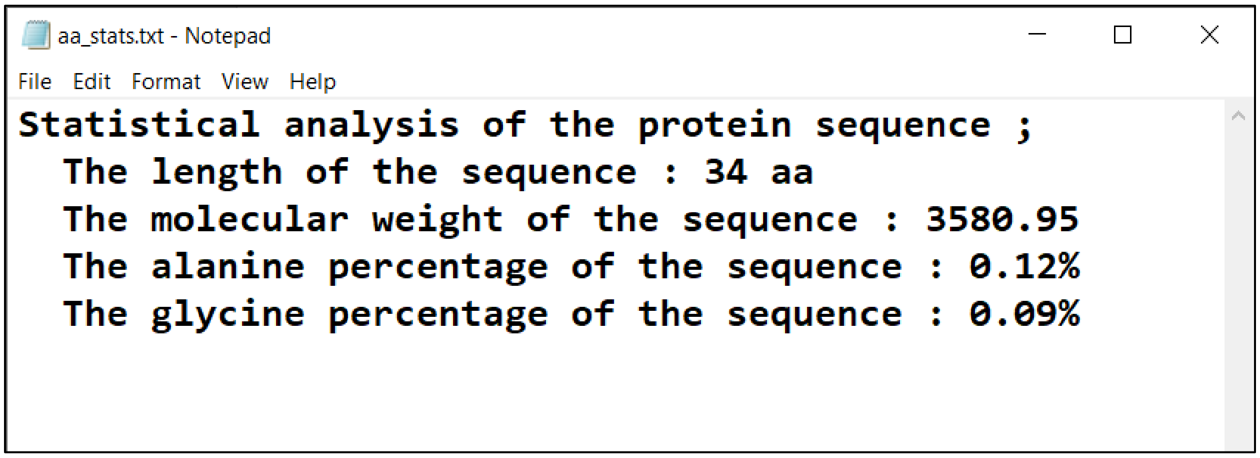
```
  GNU nano 6.2                        s14682_bi_pipeline.sh

#Simple pipeline to combine the 4 scripts in the given order
python.exe cds_seq_retrieve.py
python.exe transcribe.py
python.exe translate.py
python.exe aa_seq_analyze.py


#Create two folders
mkdir intermediate_files
mkdir output

#Move the output files to the relevent folder
mv cds_seq.fasta intermediate_files/
mv mRNA_seq.fasta intermediate_files/
mv aa_seq.fasta intermediate_files/
mv aa_stats.txt output/
```

Giving the NM_000188.3 accession number as the input;



```
nimna@LAPTOP-CGEFM648:~/s14682_02$ ./s14682_bi_pipeline.sh
Enter the accession number (with version) : NM_000188.3
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\Bio\Seq.py:3482: Biop
ythonWarning: Partial codon, len(sequence) not a multiple of three. Explicitly trim the sequ
ence or add trailing N before translation. This may become an error in future.
  warnings.warn(
nimna@LAPTOP-CGEFM648:~/s14682_02$ ls -1
aa_seq_analyze.py
cds_seq_retrieve.py
intermediate_files
output
s14682_bi_pipeline.sh
transcribe.py
translate.py
```

Output file;



```
Statistical analysis of the protein sequence ;
  The length of the sequence : 34 aa
  The molecular weight of the sequence : 3580.95
  The alanine percentage of the sequence : 0.12%
  The glycine percentage of the sequence : 0.09%
```

Output;

Statistical analysis of the protein sequence ;
  The length of the sequence : 34 aa
  The molecular weight of the sequence : 3580.95
  The alanine percentage of the sequence : 0.12%
  The glycine percentage of the sequence : 0.09%

References

- Xie, Zhouli, et al. "AP2/ERF transcription factor regulatory networks in hormone and abiotic stress responses in Arabidopsis." *Frontiers in plant science* 10 (2019): 228.