Student B number: **B236494**

Hi!
Some feedback for you for ICA2, part of which is shown here:

Tasks overview
Write a generic Python3 programme/script

- **that doesn't use BioPython**
- **that isn't just a Unix/bash script called from Python3...**
- **that you haven't used ChatGPT for...**
- that you will make available as a passworded `ccrypt` encrypted file in a PUBLIC repository on GitHub in your Bxxxxxx-2023 GitHub account

that will allow the user:

1. to identify a family of protein sequences from a user-defined subset of the taxonomic tree (e.g. glucose-6-phosphatase proteins from *Aves* (birds), or ABC transporters in mammals, or kinases in rodents, or adenyl cyclases in vertebrates *etc.*) that could then be processed using, for example, one or more of the EMBOSS programmes installed on the MSc server:
   - to determine, and plot, the level of protein sequence conservation across the species within that taxonomic group
   - to scan the protein sequence(s) of interest with motifs from the PROSITE database, to determine whether any known motifs (domains) are associated with this subset of sequences
   - to do any other appropriate EMBOSS (or other) analysis that you think might add relevant biological information to the outputs
2. write a "help manual" for your programme which has two sections:
   - one aimed at an "ordinary user" (non-coder)
   - one aimed at a competent Python3 code-writer

## A possible answer

Like ICA1, the emphasis of this ICA was for you to write something that is actually useful, complete with git repository, that could be used by you or others in the future.

It should use some, if not all (and more!), of the things we did in the Python part of the course, so that includes variables, lists, loops, dictionaries, functions, regex, dataframes perhaps, and so on... apart from Biopython!

## Code comments

- **Could I clone the GitHub repository asked for?**
  Yes, no problems!
- **Date/time on GitHub repo**
  ```
  git log | grep Date
  ```
  ```
  Date:   Sun Nov 19 21:16:31 2023 +0000
  ```
- **Could I decrypt the file with the password given?**
  Yes, worked fine!
- **Did the GitHub repository only include the pipeline code as a tar.gz filefile, as requested?**
  Yes
  ```
  ls -1
  ```
  ```
  all_the_things_I_did_for_ICA2
  B236494-2023.ICA2.tar.gz
  final_full_script_ICA2.py
  ```
- **Was your user name visible anywhere?**
  ```
  grep Author *things* | cut -d ' ' -f2,3 | sort | uniq
  ```
  ```
  git log | grep Au
  ```

  No, couldn't find it at all: well done!
- **What did the git log suggest to me?**
  ```
  egrep "Date" *things* | awk '{if(NF>1){print $2,$3,$4}}' | uniq -c
  ```
  ```
  egrep "Date" *things* | awk '{if(NF>1){print $0}}' | uniq -c
  ```
  ```
  egrep -v "Auth|Date|commit" *things* | awk '(NF>=1)'
  ```

  ### Dates

  ```
    1 Date:   Sun Nov 19 20:53:00 2023 +0000
    1 Date:   Sun Nov 19 19:03:33 2023 +0000
    1 Date:   Sun Nov 19 03:56:13 2023 +0000
    1 Date:   Sat Nov 18 23:47:40 2023 +0000
    1 Date:   Sat Nov 18 07:06:04 2023 +0000
    1 Date:   Thu Nov 16 18:58:39 2023 +0000
    1 Date:   Wed Nov 15 00:00:56 2023 +0000
    1 Date:   Mon Nov 13 23:36:33 2023 +0000
    1 Date:   Sat Nov 11 23:34:57 2023 +0000
    1 Date:   Fri Nov 10 20:59:11 2023 +0000
    1 Date:   Thu Nov 9 21:31:12 2023 +0000
    1 Date:   Tue Nov 7 22:50:30 2023 +0000
    1 Date:   Mon Nov 6 23:36:20 2023 +0000
    1 Date:   Sun Nov 5 17:11:15 2023 +0000
    1 Date:   Thu Nov 2 19:07:06 2023 +0000
  ```

  ### Messages

  ```
  Minor Correction
  Added the code for the wild card program. Finished the script. Proceeding to add to github for submission.
  Added the list of inputs and outputs of the Programme which help understand what it can and cannot do
  Added the code for the second main processing stage where we use patmatmotifs to search for motifs of interest from the PROSITE database from our subset of sequences that the u
  Added the code for one of the main processing stages where we use clustalo and plotcon to determine and visualise the level of conservation between the dataset protein sequences,
  Added functionality in the code to check the number of sequences that were obtained with the inputs, if it exceeded 1000, limit was set to 1000. If sequences obtained was only 0 or
  Asked the user if they are satisfied with the dataset, gave them the option to regenerate a new one by going back to the input section, generated a fasta file needed for further proce
  Made some minor corrections in code to further facilitate error trapping
  Wrote code to generate an ordered list of species which contain the protein sequence and is part of the specified taxon group, sorted in descending order of their frequency
  Wrote functions to accept the next input i.e. the protein sequence and displayed both the inputs to the user before proceeding to the next step
  ```

Wrote functions to accept the input taxonomic group and provide some details about the taxonomic group chosen, gave the user choice to proceed to next input
Added more notes to describe the code blocks, initiated code block to accept input from user
Made the code more readable, started adding notes
Gave the user a choice to list the features of the programme
Adding the ICA2 python3 script file to the Repository

Excellent. Coding was done over multiple sessions, good use of messages.

-
  ```
  # System check
  python3 --version
  Python 3.8.10
  which python3
  /usr/bin/python3
  ```
- **What modules were used?**
  ```
  cat *.py | grep "import " | sort | uniq -c
        1 from matplotlib import image as mpimg
        1 from matplotlib import pyplot as plt
        1 import os, sys, subprocess, time, shutil
  ```
- **Code comments: I tried to run the programme on the `msc8 server` without consulting the user / maintenance manual**
  BECAUSE THAT IS WHAT MOST USERS WILL DO...!

  ```
  # Command line versions of example runs done 20 Nov 2023
  esearch -db protein -query 'Aves[organism] AND glucose-6-phosphatase[protein name] NOT partial'
  <ENTREZ_DIRECT>
    <Db>protein</Db>
    <WebEnv>MCID_637e7ed1046db471fd3c3a3b</WebEnv>
    <QueryKey>1</QueryKey>
    <Count>61</Count>
    <Step>1</Step>
  </ENTREZ_DIRECT>
  esearch -db protein -query 'Ascomycota[organism] AND pyruvate dehydrogenase[protein name] NOT partial'
  <ENTREZ_DIRECT>
    <Db>protein</Db>
    <WebEnv>MCID_655bb7354a2fb649ec525ed3</WebEnv>
    <QueryKey>1</QueryKey>
    <Count>223</Count>
    <Step>1</Step>
  </ENTREZ_DIRECT>
  ```

  ```
  # A large dataset (Eukaryota kinase) txid2759
  esearch -db protein -query 'Eukaryota[organism] AND kinase[protein name] NOT partial'
  <ENTREZ_DIRECT>
    <Db>protein</Db>
    <WebEnv>MCID_637e7ed1046db471fd3c3a3b</WebEnv>
    <QueryKey>1</QueryKey>
    <Count>2666</Count>
    <Step>1</Step>
  </ENTREZ_DIRECT>
  ```
- **Could I run the script directly as a script (did it have a shebang line?)?**
  No, it fell over, and I had to fix it to make it work (problem was inability to make a folder if it already existed)
  ```
  # Run 1
  python3 final_full_script_ICA2.py


  Welcome to the Python3 Programme

  This programme will require 2 inputs from the User and will process and generate outputs depending on the input

  Would you like to view what you will have to input and the list of outputs that will be generated?

  Type y if you wish to do so OR n if you don't want to/already know what this programme does

  What is your choice (y/n)?      (y=yes and n=no)
  y

  Here's the list of inputs required & outputs that will be generated

  Inputs: (All choices are in the format y/n)      (Where y-=yes and n=no)

            Input Section:
            Taxonomic Group
            Protein Name,
            Choices to proceed or not (multiple)


            1st Processing Section:
            Choice to plot level of conservation of dataset protein sequences,
            winsize parameter to plot conservation level,
            choice to generate multiple images


            2nd Processing Section:
  ```

```
            Choice to begin search for motifs

Outputs:

            Input Section:
            .txt file containing info about input taxon group,
            .txt file containing ordered list of number of individual species in the dataset,
            .fasta file of sequences in the dataset


            1st Processing Section: (Programs Used => Clustalo, Plotcon)
            .aln file containing multiple sequence alignment information of the sequences,
            .png images of level of conservation according to specified winsize (can be multiple)


            2nd Processing Section: (Programs Used => Patmatmotifs)
            Folder containing each individual sequence of the fasta file generate for the dataset
            Folder containing the output file of searching for motifs in each sequence
            .txt file containing the combined outputs of running patmatmotifs on all sequences in the dataset
            .txt file containing ordered list of number of motifs found for the dataset

            We also have a wildcard at the end which is a surprise :P



The current working directory is  /localdisk/home/aivens2/AY23/marking/BPSM/ICA2/cloned_B236494/Python3_Programme_Files/

For each dataset generated the corresponding output will be be stored in a folder named as follows: (protein_name)_(taxonomic_group)_f


### INPUT SECTION OF THE CODE ###


Welcome to the input section of the code where we generate the dataset that will be used in the processing stages that follow

We require the protein to be searched for and the taxonomic group to be searched in as the 2 inputs

What is the taxonomic group that you want to query in

aves

This is the taxonomic group you have chosen and a few details about it.

TaxId   ScientificName  GenbankCommonName       Division

8782    Aves    birds   Vertebrates


WARNING! Choosing n=no for the next choice will terminate the program

Shall we proceed to input the protein (y/n)?    (y=yes and n=no)
y

You have chosen to proceed

What is the protein that you want to query with

glucose-6-phosphatase

The protein that you have chosen to query with is  glucose-6-phosphatase

The input taxonomic group is  aves  & the input protein is  glucose-6-phosphatase

Before we proceed to view the list of species that will be obtained for the inputs we first need to check what the total number of sequence

This is crucial because if we exceed 1000 sequences it will be very time consuming. So the maximum allowable sequences limit is set to 1

We also cannot do alignment with just 1 or 0 sequences, so in these scenarios we need to generate new dataset

The number of sequences obtained was:  61

WARNING! Choosing n=no for the next choice will terminate the program


Shall we proceed to view the list of species which contains our protein (y/n)?          (y=yes and n=no)
y

You have chosen to proceed


If there were close to 1000 sequences this might take a while .... You might want to get some coffee...


This is the ordered list of species which contain the input protein sorted in descending order of their frequency

      2 Lonchura striata domestica
      2 Colius striatus
      2 Calypte anna
      1 Willisornis vidua
      1 Turdus rufiventris
      1 Tauraco erythrolophus
      1 Sturnus vulgaris
      1 Strigops habroptila
      1 Pygoscelis adeliae
      1 Pseudopodoces humilis
      1 Pitangus sulphuratus
      1 Pipra filicauda
      1 Phasianus colchicus
      1 Phaethon lepturus
      1 Patagioenas fasciata monilis
      1 Passer montanus
      1 Parus major
      1 Opisthocomus hoazin
      1 Numida meleagris
      1 Nothoprocta perdicaria
      1 Nipponia nippon
      1 Neopelma chrysocephalum
      1 Motacilla alba alba
      1 Mesitornis unicolor
      1 Merops nubicus
      1 Melopsittacus undulatus
      1 Meleagris gallopavo
      1 Manacus vitellinus
      1 Limosa lapponica baueri
      1 Lepidothrix coronata
      1 Lamprotornis superbus
      1 Haliaeetus leucocephalus
      1 Geospiza fortis
      1 Ficedula albicollis
      1 Falco rusticolus
      1 Falco naumanni
      1 Empidonax traillii
      1 Dromaius novaehollandiae
      1 Cygnus olor
      1 Cyanistes caeruleus
      1 Coturnix japonica
      1 Corvus moneduloides
      1 Corvus cornix cornix
      1 Corvus brachyrhynchos
      1 Corapipo altera
      1 Columba livia
      1 Chiroxiphia lanceolata
      1 Charadrius vociferus
      1 Camarhynchus parvulus
      1 Calidris pugnax
      1 Athene cunicularia
      1 Apteryx mantelli mantelli
      1 Aptenodytes forsteri
      1 Apaloderma vittatum
      1 Anas platyrhynchos
      1 Amazona aestiva
      1 Aix galericulata
      1 Acanthisitta chloris


 This is the ordered list of species which contain the input protein sorted in descending order of their frequency


 The list is stored in 'glucose-6-phosphatase_aves.txt'


 A fasta file will now be generated for the specified inputs, which will be used for further processing


 Processing.....Processing.....Processing


 The name of the fasta file is 'glucose-6-phosphatase_aves.fasta'


 Is the current dataset acceptable? Would you like to continue to the processing stages or go back to the input stage and generate a new d

 Choices: y/yes= Continue with the current dataset ; n/no means you want to generate a new dataset


 Shall we proceed to the processing stages with the current dataset (y/n)?      (y=yes and n=no)
 y
 You have chosen to proceed with the current dataset


 A new working directory called  xxx/Python3_Programme_Files/glucose-6-phosphatase_aves_files  has been created


 All files will be moved to the new directory and it will be set as the new current working directory for the upcoming stages

 Moved:  glucose-6-phosphatase_aves.txt

 Moved:  glucose-6-phosphatase_aves.fasta

 Moved:  aves.txt


 All the files have been moved


 ### PROCESSING SECTION OF THE CODE: Level of conservation between the protein sequences ###


 Welcome to the Processing stages where we will have some interesting outputs and inferences


 We will first determine and plot the level of conservation of the dataset protein sequences

 We will be using Clustal Omega (aka clustalo) to both cluster and perform multiple sequence alignment on the dataset protein sequences


 Here we go ... Initiating alignment ...

Using 100 threads
seq-type = UNKNOWN
seq-in-fmt = unknown
```

```
option: seq-in = glucose-6-phosphatase_aves.fasta
option: dealign = 0
option: profile1 = (null)
option: profile2 = (null)
option: is-profile = 0
option: max-num-seq = 2147483647
option: max-seq-len = 2147483647
option: aln-out-file = glucose-6-phosphatase_aves_alignment_fasta.aln
option: aln-out-format = FASTA
option: force-file-overwrite = 1
option: line wrap = 60
option: print residue numbers = 0
option: order alignment like input/tree = 0
option: threads = 100
option: PseudoFile = (null)
option: logFile = (null)
option: auto-options = 0
option: distmat-infile = (null)
option: distmat-outfile = (null)
option: clustering-type = 1
option: pair-dist-type = 1
option: use-mbed = 0
option: use-mbed-for-iteration = 0
option: pile-up = 0
option: guidetree-outfile = (null)
option: guidetree-infile = (null)
option: hmm-input-files = 0
option: num-iterations = 1
option: iterations-auto = 0
option: max-hmm-iterations = 2147483647
option: max-guidetree-iterations = 2147483647
option: iMacRamMB = 8000
option: percent-id = 0
option: use-kimura = 1
option: clustering-out = (null)
option: posterior-out = (null)
Read 61 sequences (type: Protein) from glucose-6-phosphatase_aves.fasta
not more sequences (61) than cluster-size (100), turn off mBed
Calculating pairwise ktuple-distances...
Ktuple-distance calculation progress: 0 % (0 out of 1891)
Ktuple-distance calculation progress: 1 % (19 out of 1891)
Ktuple-distance calculation progress: 2 % (38 out of 1891)
...
Ktuple-distance calculation progress: 191 % (3618 out of 1891)
Ktuple-distance calculation progress: 192 % (3633 out of 1891)
Ktuple-distance calculation progress done. CPU time: 2.01u 0.05s 00:00:02.05 Elapsed: 00:00:01
Guide-tree computation done.
Progressive alignment progress: 1 % (1 out of 60)
Progressive alignment progress: 3 % (2 out of 60)
...
Progressive alignment progress: 98 % (59 out of 60)
Progressive alignment progress: 100 % (60 out of 60)
Progressive alignment progress done. CPU time: 21.61u 0.01s 00:00:21.62 Elapsed: 00:00:00
Iteration step 1 out of 1
Computing new guide tree (iteration step 32785)
Calculating Kimura-corrected pairwise aligned identity distances...
Pairwise distance calculation progress: 0 % (0 out of 1891)
Pairwise distance calculation progress: 0 % (0 out of 1891)
...
Pairwise distance calculation progress: 191 % (3618 out of 1891)
Pairwise distance calculation progress: 192 % (3633 out of 1891)
Pairwise identity calculation progress done. CPU time: 0.98u 0.00s 00:00:00.98 Elapsed: 00:00:00
Guide-tree computation done.
Computing HMM from alignment
Progressive alignment progress: 1 % (1 out of 60)
Progressive alignment progress: 3 % (2 out of 60)
...
Progressive alignment progress: 98 % (59 out of 60)
Progressive alignment progress: 100 % (60 out of 60)
Progressive alignment progress done. CPU time: 51.36u 0.09s 00:00:51.45 Elapsed: 00:00:03
Alignment written to glucose-6-phosphatase_aves_alignment_fasta.aln

 The output file is 'glucose-6-phosphatase_aves_alignment_fasta.aln'


 Let's now plot the level of conservation for the aligned sequences using plotcon


 To do this we have given you the option to choose the winsize


 A large window (e.g. 100) gives a nice, smooth curve, and very low 'similarity score' units, whereas a small window (e.g. 4) gives a very spi


 Shall we plot the level of conservation for these sequences (y/n)?      (y=yes and n=no)
y

 What is the winsize you want to choose? (please input an integer value)
10
 You have chosen a winsize of '10'


 You have chosen to proceed

Created glucose-6-phosphatase_aves_alignment_fasta_10.aln.1.png

 The image that has been generated is 'glucose-6-phosphatase_aves_alignment_fasta_10.aln.1.png'


 A beautiful graph is headed your way! Make sure to close it once you have finished viewing to continue with the Programme
 Don't worry about the plots, we are saving each and every one of them that you create


 Please give it a few seconds, the plot is loading ...

Traceback (most recent call last):
  File "/home/aivens2/.local/lib/python3.8/site-packages/matplotlib/backends/backend_gtk3.py", line 178, in key_press_event
    key = self._get_key(event)
  File "/home/aivens2/.local/lib/python3.8/site-packages/matplotlib/backends/backend_gtk3.py", line 217, in _get_key
    key = cbook._unikey_or_keysym_to_mplkey(
  File "/home/aivens2/.local/lib/python3.8/site-packages/matplotlib/cbook/__init__.py", line 2300, in _unikey_or_keysym_to_mplkey
    key = keysym.lower()
AttributeError: 'NoneType' object has no attribute 'lower'
Traceback (most recent call last):
  File "/home/aivens2/.local/lib/python3.8/site-packages/matplotlib/backends/backend_gtk3.py", line 183, in key_release_event
    key = self._get_key(event)
  File "/home/aivens2/.local/lib/python3.8/site-packages/matplotlib/backends/backend_gtk3.py", line 217, in _get_key
    key = cbook._unikey_or_keysym_to_mplkey(
  File "/home/aivens2/.local/lib/python3.8/site-packages/matplotlib/cbook/__init__.py", line 2300, in _unikey_or_keysym_to_mplkey
    key = keysym.lower()
 AttributeError: 'NoneType' object has no attribute 'lower'

 The plot has been saved as 'glucose-6-phosphatase_aves_alignment_fasta_10.aln.1.png'


 Do you want to generate another image with a different winsize (y/n)?   (y=yes and n=no)
y

 You have decided to generate another image


 Shall we plot the level of conservation for these sequences (y/n)?      (y=yes and n=no)
y

 What is the winsize you want to choose? (please input an integer value)
 25
 You have chosen a winsize of '25'


 You have chosen to proceed

Created glucose-6-phosphatase_aves_alignment_fasta_25.aln.1.png

 The image that has been generated is 'glucose-6-phosphatase_aves_alignment_fasta_25.aln.1.png'


 A beautiful graph is headed your way! Make sure to close it once you have finished viewing to continue with the Programme


 Don't worry about the plots, we are saving each and every one of them that you create


 Please give it a few seconds, the plot is loading ...
 n

 You have decided not to generate another image


 I hope you liked the images that were generated !!!


 ### PROCESSING SECTION OF THE CODE: Scanning for Motifs in the protein sequences ###


 Welcome to the Motif searching section of this Programme

 We will be searcing for motifs in the PROSITE database using the sequences in the dataset

 Are you ready to do some motif searching ?


 WARNING! Choosing n=no for the next choice will terminate the program

 Shall we proceed to search for the motifs (y/n)?        (y=yes and n=no)
y

 You have chosen to proceed


 The fasta file we will be using to scan is  glucose-6-phosphatase_aves.fasta

 Fasta File copied successfully to be used for searching


 All the fasta files have been stored in glucose-6-phosphatase_aves_individual_fasta_files


 Time to start the search ... Hold on to your seat ... This is fast and furious ...

Scan a protein sequence with motifs from the PROSITE database
Scan a protein sequence with motifs from the PROSITE database
...
Scan a protein sequence with motifs from the PROSITE database

 All the patmatmotif files have been stored in patmatmotif_outputs


 All patmatmotif results are combined into glucose-6-phosphatase_aves_combined_patmatmotifs_results.txt
```

```
Motifs present have been successfully extracted and saved to 'motifs_present_in_glucose-6-phosphatase_aves.txt'


These are the motifs that were present in our database:

    52  AMIDATION



When we run backtranseq on our dataset we get nucleic acid sequences that the proteins in our sequences most likely came from

Back-translate a protein sequence to a nucleotide sequence

Process Completed


The file containing the result is stored in glucose-6-phosphatase_aves_nucleotide.txt', if you want to view it


       Final Message:
       We have come to the END OF THIS PROGRAMME.
       I hope you have gained some useful output from this programme.
       Please view the outputs generated again if you wish to do so.
       Thank you for trying this programme, refer the help manual for more information on how to best run this programme.
       Fin.
```

REASON: finished OK!

```
# What files do we have?
ls -R
```

```
.:
all_the_things_I_did_for_ICA2  B236494-2023.ICA2.tar.gz  final_full_script_ICA2.py  Python3_Programme_Files

./Python3_Programme_Files:
glucose-6-phosphatase_aves_files

./Python3_Programme_Files/glucose-6-phosphatase_aves_files:
aves.txt                                        glucose-6-phosphatase_aves_alignment_fasta.aln  motifs_search_PROSITE_db
glucose-6-phosphatase_aves_alignment_fasta_10.aln.1.png  glucose-6-phosphatase_aves.fasta
glucose-6-phosphatase_aves_alignment_fasta_25.aln.1.png  glucose-6-phosphatase_aves.txt

./Python3_Programme_Files/glucose-6-phosphatase_aves_files/motifs_search_PROSITE_db:
glucose-6-phosphatase_aves_combined_patmatmotifs_results.txt  glucose-6-phosphatase_aves_nucleotide.txt
glucose-6-phosphatase_aves.fasta                              motifs_present_in_glucose-6-phosphatase_aves.txt
glucose-6-phosphatase_aves_individual_fasta_files             patmatmotif_outputs

./Python3_Programme_Files/glucose-6-phosphatase_aves_files/motifs_search_PROSITE_db/glucose-6-phosphatase_aves_individual_fasta_fil
seq_10.fasta  seq_16.fasta  seq_21.fasta  seq_27.fasta  seq_32.fasta  seq_38.fasta  seq_43.fasta  seq_49.fasta  seq_54.fasta  seq_5.fasta   seq_
seq_11.fasta  seq_17.fasta  seq_22.fasta  seq_28.fasta  seq_33.fasta  seq_39.fasta  seq_44.fasta  seq_4.fasta   seq_55.fasta  seq_60.fasta
seq_12.fasta  seq_18.fasta  seq_23.fasta  seq_29.fasta  seq_34.fasta  seq_3.fasta   seq_45.fasta  seq_50.fasta  seq_56.fasta  seq_61.fasta
seq_13.fasta  seq_19.fasta  seq_24.fasta  seq_2.fasta   seq_35.fasta  seq_40.fasta  seq_46.fasta  seq_51.fasta  seq_57.fasta  seq_6.fasta
seq_14.fasta  seq_1.fasta   seq_25.fasta  seq_30.fasta  seq_36.fasta  seq_41.fasta  seq_47.fasta  seq_52.fasta  seq_58.fasta  seq_7.fasta
seq_15.fasta  seq_20.fasta  seq_26.fasta  seq_31.fasta  seq_37.fasta  seq_42.fasta  seq_48.fasta  seq_53.fasta  seq_59.fasta  seq_8.fasta

./Python3_Programme_Files/glucose-6-phosphatase_aves_files/motifs_search_PROSITE_db/patmatmotif_outputs:
seq_10.fasta.patmatmotifs  seq_22.fasta.patmatmotifs  seq_34.fasta.patmatmotifs  seq_46.fasta.patmatmotifs  seq_58.fasta.patmatmotifs
seq_11.fasta.patmatmotifs  seq_23.fasta.patmatmotifs  seq_35.fasta.patmatmotifs  seq_47.fasta.patmatmotifs  seq_59.fasta.patmatmotifs
seq_12.fasta.patmatmotifs  seq_24.fasta.patmatmotifs  seq_36.fasta.patmatmotifs  seq_48.fasta.patmatmotifs  seq_5.fasta.patmatmotifs
seq_13.fasta.patmatmotifs  seq_25.fasta.patmatmotifs  seq_37.fasta.patmatmotifs  seq_49.fasta.patmatmotifs  seq_60.fasta.patmatmotifs
seq_14.fasta.patmatmotifs  seq_26.fasta.patmatmotifs  seq_38.fasta.patmatmotifs  seq_4.fasta.patmatmotifs   seq_61.fasta.patmatmotifs
seq_15.fasta.patmatmotifs  seq_27.fasta.patmatmotifs  seq_39.fasta.patmatmotifs  seq_50.fasta.patmatmotifs  seq_6.fasta.patmatmotifs
seq_16.fasta.patmatmotifs  seq_28.fasta.patmatmotifs  seq_3.fasta.patmatmotifs   seq_51.fasta.patmatmotifs  seq_7.fasta.patmatmotifs
seq_17.fasta.patmatmotifs  seq_29.fasta.patmatmotifs  seq_40.fasta.patmatmotifs  seq_52.fasta.patmatmotifs  seq_8.fasta.patmatmotifs
seq_18.fasta.patmatmotifs  seq_2.fasta.patmatmotifs   seq_41.fasta.patmatmotifs  seq_53.fasta.patmatmotifs  seq_9.fasta.patmatmotifs
seq_19.fasta.patmatmotifs  seq_30.fasta.patmatmotifs  seq_42.fasta.patmatmotifs  seq_54.fasta.patmatmotifs
seq_1.fasta.patmatmotifs   seq_31.fasta.patmatmotifs  seq_43.fasta.patmatmotifs  seq_55.fasta.patmatmotifs
seq_20.fasta.patmatmotifs  seq_32.fasta.patmatmotifs  seq_44.fasta.patmatmotifs  seq_56.fasta.patmatmotifs
seq_21.fasta.patmatmotifs  seq_33.fasta.patmatmotifs  seq_45.fasta.patmatmotifs  seq_57.fasta.patmatmotifs
```

```
# TOO many pauses....!
grep -c time final_full_script_ICA2.py
```

```
95
```

```
perl -pi -e 's/time/\#time/g' final_full_script_ICA2.py
```

```
# Run 2
python3 final_full_script_ICA2.py
```

```
Traceback (most recent call last):
  File "final_full_script_ICA2.py", line 17, in <module>
    os.mkdir(main_pwd_to_be_set)
FileExistsError: [Errno 17] File exists: '/xxx/Python3_Programme_Files/'
```

REASON: `makedirs()` **should be used here to prevent this**

```
# Run 3
```

```
mv ./Python3_Programme_Files/ ./Run1_Python3_Programme_Files
```

```
python3 final_full_script_ICA2.py
```

```

 Welcome to the Python3 Programme

 This programme will require 2 inputs from the User and will process and generate outputs depending on the input

 Would you like to view what you will have to input and the list of outputs that will be generated?

 Type y if you wish to do so OR n if you don't want to/already know what this programme does

 What is your choice (y/n)?      (y=yes and n=no)
 n

 You have chosen not to list the inputs & outputs


 The current working directory is  /localdisk/home/aivens2/AY23/marking/BPSM/ICA2/cloned_B236494/Python3_Programme_Files/

 For each dataset generated the corresponding output will be be stored in a folder named as follows: (protein_name)_(taxonomic_group)_f


 ### INPUT SECTION OF THE CODE ###


 Welcome to the input section of the code where we generate the dataset that will be used in the processing stages that follow

 We require the protein to be searched for and the taxonomic group to be searched in as the 2 inputs

 What is the taxonomic group that you want to query in
 ascomycetes

 This is the taxonomic group you have chosen and a few details about it.

TaxId   ScientificName  GenbankCommonName       Division

4890    Ascomycota      ascomycetes     Plants and Fungi


 WARNING! Choosing n=no for the next choice will terminate the program

 Shall we proceed to input the protein (y/n)?    (y=yes and n=no)
pyruvate dehydrogenase

 The protein that you have chosen to query with is  pyruvate_dehydrogenase

 The input taxonomic group is  ascomycetes  & the input protein is  pyruvate_dehydrogenase

 Before we proceed to view the list of species that will be obtained for the inputs we first need to check what the total number of sequence

 This is crucial because if we exceed 1000 sequences it will be very time consuming. So the maximum allowable sequences limit is set to 1

 We also cannot do alignment with just 1 or 0 sequences, so in these scenarios we need to generate new dataset


 The number of sequences obtained was:  223

 WARNING! Choosing n=no for the next choice will terminate the program


 Shall we proceed to view the list of species which contains our protein (y/n)?          (y=yes and n=no)
 y

 You have chosen to proceed

g
 If there were close to 1000 sequences this might take a while .... You might want to get some coffee...


 This is the ordered list of species which contain the input protein sorted in descending order of their frequency

      9 Histoplasma capsulatum G186AR
      7 Aspergillus flavus
      6 Histoplasma capsulatum
      6 Aspergillus neoniger CBS 115656
      6 Aspergillus costaricaensis CBS 115574
      4 Saccharomyces cerevisiae
      4 Cordyceps javanica
      4 Colletotrichum tofieldiae
      4 Beauveria bassiana ARSEF 2860
      4 Aspergillus vadensis CBS 113365
      4 Aspergillus tubingensis
      4 Aspergillus piperis CBS 112811
      4 Aspergillus luchuensis
      4 Aspergillus eucalypticola CBS 122712
      3 Talaromyces pinophilus
      3 Spathaspora sp. JA1
      3 Meyerozyma sp. JA9
      3 Histoplasma ohiense (nom. inval.)
      3 Histoplasma capsulatum var. duboisii H88
      3 Aspergillus niger
      2 Yamadazyma tenuis ATCC 10573
      2 Xylona heveae TC161
      2 Trichoderma lentiforme
      2 Trichoderma citrinoviride
      2 Trematosphaeria pertusa
      2 Sporothrix schenckii 1099-18
      2 Sporothrix brasiliensis 5110
```

```
     2 Saitoella complicata NRRL Y-17804
     2 Penicillium ucsense
     2 Penicillium rolfsii
     2 Penicillium macrosclerotiorum
     2 Penicillium chrysogenum
     2 Nemania serpens
     2 Mollisia scopiformis
     2 Metschnikowia bicuspidata var. bicuspidata NRRL YB-4993
     2 Metarhizium robertsii ARSEF 23
     2 Lindgomyces ingoldianus
     2 Grosmannia clavigera kw1407
     2 Fusarium oxysporum
     2 Fusarium culmorum
     2 Drepanopeziza brunnea f. sp. 'multigermtubi' MB_m1
     2 Drechmeria coniospora
     2 Daldinia caldariorum
     2 Cyberlindnera jadinii NRRL Y-1542
     2 Colletotrichum truncatum
     2 Colletotrichum higginsianum IMI 349063
     2 Aspergillus steynii IBT 23096
     2 Aspergillus sp. HF37
     2 Aspergillus sclerotialis
     2 Aspergillus novofumigatus IBT 16806
     2 Aspergillus nomiae NRRL 13137
     2 Aspergillus niger CBS 101883
     2 Aspergillus luchuensis IFO 4308
     2 Aspergillus heteromorphus CBS 117.55
     2 Aspergillus fischeri NRRL 181
     2 Aspergillus clavatus NRRL 1
     2 Aspergillus bombycis
     2 Aspergillus aculeatinus CBS 121060
     2 Ascoidea rubescens DSM 1968
     1 Wilcoxina mikolae CBS 423.85
     1 Tuber magnatum
     1 Trichoderma longibrachiatum ATCC 18648
     1 Tolypocladium paradoxum
     1 Thozetella sp. PMI_491
     1 Terfezia boudieri ATCC MYA-4762
     1 Sclerotinia borealis F-4128
     1 Schizothecium conicum
     1 Polyplosphaeria fusca
     1 Podospora appendiculata
     1 Plectosphaerella plurivora
     1 Plectosphaerella cucumerina
     1 Pichia membranifaciens
     1 Periconia macrospinosa
     1 Penicillium longicatenatum
     1 Penicillium brasilianum
     1 Ophiostoma piceae UAMH 11346
     1 Niveomyces insectorum RCEF 264
     1 Nemania abortiva
     1 Myriangium duriaei CBS 260.36
     1 Morchella conica CCBAS932
     1 Metarhizium guizhouense ARSEF 977
     1 Lophium mytilinum
     1 Leptodontidium sp. MPI-SDFR-AT-0119
     1 Lepidopterella palustris CBS 459.81
     1 Kluyveromyces marxianus
     1 Hypoxylon sp. NC0597
     1 Hypoxylon sp. FL1857
     1 Hypoxylon sp. FL0890
     1 Hypoxylon sp. FL0543
     1 Hypoxylon sp. EC38
     1 Hanseniaspora valbyensis NRRL Y-1626
     1 Halenospora varia
     1 Glonium stellatum
     1 Didymosphaeria enalia
     1 Decorospora gaudefroyi
     1 Cordyceps militaris
     1 Coniochaeta sp. PMI_546
     1 Coniochaeta ligniaria NRRL 30616
     1 Colletotrichum incanum
     1 Colletotrichum higginsianum
     1 Colletotrichum asianum
     1 Choiromyces venosus 120613-1
     1 Chaetomium fimeti
     1 Cercophora scortea
     1 Cenococcum geophilum 1.58
     1 Candida parapsilosis
     1 Beauveria bassiana D1-5
     1 Aspergillus ustus
     1 Aspergillus sclerotiicarbonarius CBS 121057
     1 Aspergillus phoenicis ATCC 13157
     1 Aspergillus niger ATCC 13496
     1 Aspergillus fumigatus Z5
     1 Aspergillus carlsbadensis
     1 Aspergillus arachidicola
     1 Ascodesmis nigricans
     1 Ascobolus immersus RN42


This is the ordered list of species which contain the input protein sorted in descending order of their frequency


The list is stored in 'pyruvate_dehydrogenase_ascomycetes.txt'


A fasta file will now be generated for the specified inputs, which will be used for further processing


Processing.....Processing.....Processing
```

```
The name of the fasta file is 'pyruvate_dehydrogenase_ascomycetes.fasta'


Is the current dataset acceptable? Would you like to continue to the processing stages or go back to the input stage and generate a new d

Choices: y/yes= Continue with the current dataset ; n/no means you want to generate a new dataset


Shall we proceed to the processing stages with the current dataset (y/n)?       (y=yes and n=no)
....
The file containing the result is stored in pyruvate_dehydrogenase_ascomycetes_nucleotide.txt', if you want to view it


      Final Message:
      We have come to the END OF THIS PROGRAMME.
      I hope you have gained some useful output from this programme.
      Please view the outputs generated again if you wish to do so.
      Thank you for trying this programme, refer the help manual for more information on how to best run this programme.
      Fin.

REASON: finished OK!
```

```
# Run 4
mv ./Python3_Programme_Files/ ./Run2_Python3_Programme_Files
python3 final_full_script_ICA2.py
```

```
The protein that you have chosen to query with is  kinase

 The input taxonomic group is  eukaryota  & the input protein is  kinase

 Before we proceed to view the list of species that will be obtained for the inputs we first need to check what the total number of sequence

 This is crucial because if we exceed 1000 sequences it will be very time consuming. So the maximum allowable sequences limit is set to 1

 We also cannot do alignment with just 1 or 0 sequences, so in these scenarios we need to generate new dataset


 The number of sequences obtained was:  1000

 WARNING! Choosing n=no for the next choice will terminate the program


 Shall we proceed to view the list of species which contains our protein (y/n)?        (y=yes and n=no)
 n

 You have chosen not to proceed

 Terminating program, This is goodbye ;_;

REASON: it wont let me do more than 1000
```

```
# Run 5: run with something that gives no proteins...
# These are MY results
```

source

```
esearch -db protein -query 'Klingon[organism] AND green slime[protein name] NOT partial'
```

```
WARNING:  FAILURE ( Tues 20 Nov 20:15:14 GMT 2023 )
nquire -url https://eutils.ncbi.nlm.nih.gov/entrez/eutils/ esearch.fcgi -retmax 0 -usehistory y -db protein -term "Klingon [organism] AND gr
<ErrorList>
  <PhraseNotFound>Klingon[organism]</PhraseNotFound>
  <PhraseNotFound>green slime[protein name]</PhraseNotFound>
<ErrorList>
SECOND ATTEMPT
 WARNING:  FAILURE ( Tues 20 Nov 20:15:16 GMT 2023 )
nquire -url https://eutils.ncbi.nlm.nih.gov/entrez/eutils/ esearch.fcgi -retmax 0 -usehistory y -db protein -term "Klingon [organism] AND gr
<ErrorList>
  <PhraseNotFound>Klingon[organism]</PhraseNotFound>
  <PhraseNotFound>green slime[protein name]</PhraseNotFound>
<ErrorList>
LAST ATTEMPT
 ERROR:  FAILURE ( Tues 20 Nov 20:15:17 GMT 2023 )
nquire -url https://eutils.ncbi.nlm.nih.gov/entrez/eutils/ esearch.fcgi -retmax 0 -usehistory y -db protein -term "Klingon [organism] AND gr
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE eSearchResult>
<eSearchResult>
  <Count>0</Count>
  <RetMax>0</RetMax>
  <RetStart>0</RetStart>
  <QueryKey>1</QueryKey>
  <WebEnv>MCID_637e7f5635097c618c5a7faf</WebEnv>
  <QueryTranslation>Klingon[organism] AND green slime[protein name] NOT partial[All Fields]</QueryTranslation>
  <ErrorList>
    <PhraseNotFound>Klingon[organism]</PhraseNotFound>
    <PhraseNotFound>green slime[protein name]</PhraseNotFound>
  </ErrorList>
  <WarningList>
    <OutputMessage>No items found.</OutputMessage>
  </WarningList>
</eSearchResult>
QUERY FAILURE
<ENTREZ_DIRECT>
  <Db>protein</Db>
  <WebEnv>MCID_637e7f58d1b1ae6f00693445</WebEnv>
  <QueryKey>1</QueryKey>
  <Count>0</Count>
  <Step>1</Step>
</ENTREZ_DIRECT>
```

```
rm -fr ./Python3_Programme_Files/
```

```
python3 final_full_script_ICA2.py
```

```
 What is the taxonomic group that you want to query in
 klingon

 This is the taxonomic group you have chosen and a few details about it.

No error trap here?

TaxId   ScientificName  GenbankCommonName       Division



 WARNING! Choosing n=no for the next choice will terminate the program

 Shall we proceed to input the protein (y/n)?    (y=yes and n=no)
 y

 You have chosen to proceed


 What is the protein that you want to query with
 green slime

 The protein that you have chosen to query with is  green_slime

 The input taxonomic group is  klingon  & the input protein is  green_slime

 Before we proceed to view the list of species that will be obtained for the inputs we first need to check what the total number of sequence

 This is crucial because if we exceed 1000 sequences it will be very time consuming. So the maximum allowable sequences limit is set to 1

 We also cannot do alignment with just 1 or 0 sequences, so in these scenarios we need to generate new dataset

 Insufficient number of sequences to proceed to processing stage.

 Number of sequences obtained:  0

 Please generate a new dataset with different inputs to proceed

 Your current inputs were  klingon  &  green_slime for reference

 Sending you back in time to provide the inputs again


 ### INPUT SECTION OF THE CODE ###


 Welcome to the input section of the code where we generate the dataset that will be used in the processing stages that follow

 We require the protein to be searched for and the taxonomic group to be searched in as the 2 inputs

 What is the taxonomic group that you want to query in

Reason: killed as there were no sequences
 ^CTraceback (most recent call last):
  File "final_full_script_ICA2.py", line 152, in
    taxo_group = input(" What is the taxonomic group that you want to query in \n ").replace(" ","_").lower()
KeyboardInterrupt
```

- **Did the script ask/tell me where I wanted to put files while it was running/when it was finished?**
  Yes, good, but it fell over the second time, as the directory already existed...
- **Could I run the script directly in my home space (i.e. no links to places I can't get to easily without immediately knowing who you are)?**
  Yes, seemed to find everything.
- **Was it actually a Python programme as requested, and not just a short Python wrapper for a Unix script?**
  Yes, definitely Python
- **Did the code have lots of comments so I could see what the code was doing?**
  Yes, more than sufficient, excellent in fact!
- **Was it logically laid out?**
  Yes
  Yes, but the user has to run several scripts; could they not be linked by, for example, an `import` statement to initiate the next step by default? That still leaves functionality/flexibility should it be be needed!
  Not really, essentially all linear, no functions?
- **Was there evidence that a flow-chart or similar had been used in planning the design (i.e. output from one bit being the input for the next)?**
  Yes
- **Were there any errors, and if so, what caused them?**
  Did not break under normal use, but I WAS able to break it by running it more than once...

- **Which components of python were used?**

```
../grepcheck | awk '{gsub("\t","",$0); gsub("  "," ",$0); gsub("  "," ",$0); print $0}' > gcheck.out
cat gcheck.out
```

```
# Looking for import statements
3
  1 from matplotlib import image as mpimg
  1 from matplotlib import pyplot as plt
  1 import os, sys, subprocess, time, shutil
```

```
# Looking for SeqIO|biopython
0
```

```
# Looking for os
29
  1 current_wd = os.getcwd()
  1 destination_file_location = os.path.join(dir_for_prosite, fasta_file_name)
  1 destination = os.path.join(new_wd_destination, file_name)
  1 fasta_file_location = os.path.join(current_wd, fasta_file_name)
  1 file_path = os.path.join(fasta_dir, f"seq_{i}.fasta")
  1 for file in os.listdir(fasta_dir):
  1 for output_filename in os.listdir(patmatmotif_dir):
  1 if not os.path.exists(fasta_file_name):
  1 if os.path.isfile(source):
  1 individual_fasta_file_path = os.path.join(fasta_dir, file)
  1 new_wd_destination = os.path.join(source_folder_cwd, f"{protein_sequence}_{taxo_group}_files")
  1 os.chdir(dir_for_prosite)
  1 os.chdir(main_pwd_to_be_set)
  1 os.chdir(new_wd_destination)
  1 os.mkdir(dir_for_prosite)
  1 os.mkdir(fasta_dir)
  1 os.mkdir(main_pwd_to_be_set)
  1 os.mkdir(new_wd_destination)
  1 os.mkdir(patmatmotif_dir)
  1 os.system(backtranseq_command)
  1 os.system(clustalo_command)
  1 os.system(esearch_command)
  1 os.system(gen_fasta_file)
  1 os.system(plotcon_command)
  1 output_file_path = os.path.join(patmatmotif_dir, f"{file}.patmatmotifs")
  1 output_file_path = os.path.join(patmatmotif_dir, output_filename)
  1 source_folder_cwd= os.getcwd()
  1 source = os.path.join(source_folder_cwd, file_name)
  1 user_cwd = os.getcwd()
```

```
# Looking for os.system
5
  1 os.system(backtranseq_command)
  1 os.system(clustalo_command)
  1 os.system(esearch_command)
  1 os.system(gen_fasta_file)
  1 os.system(plotcon_command)
```

```
# Looking for os.remove
0
```

```
# Looking for subprocess
4
  1 subprocess.run(bash_command, shell=True)
  1 subprocess.run(esearch_command_list, shell=True, capture_output=True, check=True)
  1 subprocess.run(f"patmatmotifs -sequence {individual_fasta_file_path} -full -outfile {output_file_path}", shell=True)
  1 total_value = int(subprocess.run(total_sequence_command, shell=True, capture_output=True, text=True).stdout.strip())
```

```
# Looking for write
2
```

```
# Looking for dictionaries
0
```

```
# Looking for matplot use
3
```

```
# Looking for functions
7
    def choice_y_or_n2(list_outputs_2):
    def choice_y_or_n3():
def choice_y_or_n(list_outputs):
    def edirect_pro_seq(pro_seq):
    def edirect_taxo_name(t_g):
def plotcon_multiple_plots():
def user_input_winsize():
```

```
# Looking for for loops
12
```

```
# Looking for ifs
25
```

```
# Looking for lists
0
```

# Looking for arrays

```
0
```

# Looking for regex

```
0
```

# Looking for pandas

```
0
```

# Looking for try/except

```
0
    try:
    except ValueError:
    try:
    except:
    Thank you for trying this programme, refer the help manual for more information on how to best run this programme.
```

# Checking use of esearch

```
14
    esearch_command= (f"esearch -db taxonomy -query '{t_g}' | efetch -format xml |"
    esearch_command_list =(f"esearch -db protein -query '{protein_sequence}[Protein] AND {taxo_group}[Organism] NOT PARTIAL' |"
   f"efetch -format docsum -start 1 -stop '{total_value}' |"
 f"efetch -format fasta -start 1 -stop '{total_value}' > '{protein_sequence}_{taxo_group}.fasta'"
  f"xtract -pattern Count -element Count"
   f"xtract -pattern DocumentSummary -element Organism | sort | uniq -c | sort -nr > '{protein_sequence}_{taxo_group}.txt'"
  f"xtract -pattern Taxon -sep '|' -element TaxId ScientificName GenbankCommonName Division > '{t_g}.txt'"
    gen_fasta_file =(f"esearch -db protein -query '{protein_sequence}[Protein] AND {taxo_group}[Organism] NOT PARTIAL' |"
    total_sequence_command  = (f"esearch -db protein -query '{protein_sequence}[Protein] AND {taxo_group}[Organism] NOT PARTIAL' |"
```

# Checking use of clustalo

```
2
clustalo_command= f"clustalo -i '{protein_sequence}_{taxo_group}.fasta' --full --full-iter --iter=1 --use-kimura -o '{protein_sequence}_{taxo_group}_alignment_fasta.aln' --outfm
### NOTE_11_Details: Moving into the main processing stages. We will begin by first determining and plotting the level of conservation, we will be using clustalo to perform the
```

# Checking use of force

```
1
clustalo_command= f"clustalo -i '{protein_sequence}_{taxo_group}.fasta' --full --full-iter --iter=1 --use-kimura -o '{protein_sequence}_{taxo_group}_alignment_fasta.aln' --outfm
```

# Checking use of plotcon

```
4
def plotcon_multiple_plots():
### NOTE_12_Details: Give the user the option to generate plots of the level of conservation using plotcon according to the winsize of their specification, put this in a loop to rei
    os.system(plotcon_command)
    plotcon_command= (f"plotcon -sequences '{protein_sequence}_{taxo_group}_alignment_fasta.aln' -winsize '{winsize_input_user}' -graph png -goutfile '{protein_sequence}_{ta
plotcon_multiple_plots()
print("\n Let's now plot the level of conservation for the aligned sequences using plotcon \n")
### special parameters used for plotcon
### This function will take the user input winsize and use it's value in the -winsize parameter of plotcon ###
```

# Checking use of blastp

# Checking use of awk

# Checking use of print()

```
109
  4 print("""
  2 print("""
  1 print("""
  1 print(" Are you ready to do some motif searching ? \n")
  1 print(" Before we proceed to view the list of species that will be obtained for the inputs we first need to check what the total number of sequences obtained is \n")
  1 print(" Choices: y/yes= Continue with the current dataset ; n/no means you want to generate a new dataset \n")
  1 print(content)
  1 print(f"File {fasta_file_name} not found.")
  1 print(f" File not found: {file_name}")
  1 print(file.read())
  1 print(file.read() + "\n")
  1 print(f"\n All patmatmotif results are combined into {combined_output_filename}\n")
  1 print(f"\n All the fasta files have been stored in {fasta_dir}\n")
  1 print(f"\n All the patmatmotif files have been stored in {patmatmotif_dir}\n")
  1 print(f"\n Motifs present have been successfully extracted and saved to '{output_file_name}' \n")
  1 print(f"\n The file containing the result is stored in {protein_sequence}_{taxo_group}_nucleotide.txt', if you want to view it\n")
  1 print(f"\n The image that has been generated is '{output_image}' \n")
  1 print(f"\n The list is stored in '{protein_sequence}_{taxo_group}.txt' \n")
  1 print(f"\n The name of the fasta file is '{protein_sequence}_{taxo_group}.fasta' \n")
  1 print(f"\n The output file is '{protein_sequence}_{taxo_group}_alignment_fasta.aln' \n")
  1 print(f"\n The plot has been saved as '{output_image}' \n")
  1 print(f"\n These are the motifs that were present in our database: \n")
  1 print(f" You have chosen a winsize of '{winsize_input}' \n")
  1 print(" Inputs: (All choices are in the format y/n) \t (Where y-=yes and n=no)")
  1 print(" Insufficient number of sequences to proceed to processing stage.\n ")
  1 print(" Invalid choice: Please type y or n to proceed\n")
  1 print(" Invalid input, please enter an integer \n")
  1 print(" Moved: ", file_name,"\n")
  1 print("\n A beautiful graph is headed your way! Make sure to close it once you have finished viewing to continue with the Programme \n")
  1 print("\n A fasta file will now be generated for the specified inputs, which will be used for further processing \n")
  1 print("\n A large window (e.g. 100) gives a nice, smooth curve, and very low 'similarity score' units, whereas a small window (e.g. 4) gives a very spikey, noisy plot with 'similar
  1 print("\n All files will be moved to the new directory and it will be set as the new current working directory for the upcoming stages \n")
  1 print("\n All the files have been moved \n")
  1 print("\n A new working directory called ", new_wd_destination, " has been created \n")
  1 print("\n Don't worry about the plots, we are saving each and every one of them that you create \n")
  1 print("\n Fasta File copied successfully to be used for searching \n")
  1 print("\n For each dataset generated the corresponding output will be be stored in a folder named as follows: (protein_name)_(taxonomic_group)_files \n")
  1 print("\n Here's the list of inputs required & outputs that will be generated \n")
  1 print("\n Here we go ... Initiating alignment ... \n")
  1 print("\n If there were close to 1000 sequences this might take a while .... You might want to get some coffee... \n")
  1 print("\n I hope you liked the images that were generated !!! \n")
  1 print("\n ### INPUT SECTION OF THE CODE ### \n")
  1 print("\n Invalid choice: Please type y or n to proceed \n")
  1 print("\n Invalid choice: Please type y or n to proceed\n")
  1 print("\n Invalid choice: Please type y or n to proceed \n ")
  1 print("\n Invalid choice: Please type y or n to proceed\n")
  1 print("\n Is the current dataset acceptable? Would you like to continue to the processing stages or go back to the input stage and generate a new dataset \n")
  1 print("\n Let's now plot the level of conservation for the aligned sequences using plotcon \n")
  1 print("\n Please give it a few seconds, the plot is loading ... \n")
  1 print("\n Please input a search string \n")
  1 print("\n Please input a search string, not just numbers \n")
  1 print("\n Process Completed \n")
  1 print("\n Processing.....Processing.....Processing \n")
  1 print("\n ### PROCESSING SECTION OF THE CODE: Level of conservation between the protein sequences ### \n")
  1 print("\n ### PROCESSING SECTION OF THE CODE: Scanning for Motifs in the protein sequences ### \n")
  1 ### print("\n ",protein_sequence," \n")  ### Used to gauge the result ###
  1 print("\n The current working directory is ", main_pwd_to_be_set)
  1 print("\n The fasta file we will be using to scan is ", fasta_file_name)
  1 print("\n The input taxonomic group is ",taxo_group," & the input protein is ",protein_sequence, "\n")
  1 print("\n The number of sequences obtained was: ", total_value)
  1 print("\n The protein that you have chosen to query with is ", pro_seq)
  1 print ("\n This is the ordered list of species which contain the input protein sorted in descending order of their frequency \n")
  1 print("\n This is the ordered list of species which contain the input protein sorted in descending order of their frequency \n")
  1 print(" \n This is the taxonomic group you have chosen and a few details about it.  \n")
  1 print("\n Time to go back in time to the input section \n")
  1 print("\n Time to start the search ... Hold on to your seat ... This is fast and furious ... \n")
  1 print("\n To do this we have given you the option to choose the winsize \n")
  1 print("\n Uh-Oh, something is wrong ")
  1 print(" Number of sequences obtained: ", total_value,"\n")
  2 print("\n WARNING! Choosing n=no for the next choice will terminate the program \n")
  1 print("\n WARNING! Choosing n=no for the next choice will terminate the program \n")
  1 print("\n Welcome to the input section of the code where we generate the dataset that will be used in the processing stages that follow ")
  1 print("\n Welcome to the Motif searching section of this Programme \n")
  1 print("\n Welcome to the Processing stages where we will have some interesting outputs and inferences \n")
  1 print("\n Welcome to the Python3 Programme \n")
  1 print("\n We require the protein to be searched for and the taxonomic group to be searched in as the 2 inputs\n")
  1 print("\n We will be using Clustal Omega (aka clustalo) to both cluster and perform multiple sequence alignment on the dataset protein sequences \n")
  1 print("\n We will first determine and plot the level of conservation of the dataset protein sequences ")
  1 print("\n When we run backtranseq on our dataset we get nucleic acid sequences that the proteins in our sequences most likely came from \n")
  1 print("\n You have chosen not to list the inputs & outputs\n")
  1 print("\n You have chosen not to plot the level of conservation. Moving to next stage \n")
  1 print("\n You have chosen not to proceed\n")
  1 print("\n You have chosen to proceed\n")
  1 print("\n You have chosen to proceed\n")
  1 print("\n You have decided not to generate another image \n")
  1 print("\n You have decided to generate another image \n")
  1 print(" Outputs: ")
  1 print(" Please generate a new dataset with different inputs to proceed \n")
  1 print(" Sending you back in time to provide the inputs again \n")
  1 print("TaxId\tScientificName\tGenbankCommonName\tDivision \n")
  1 ### print(" ",taxo_group) ### Used to gauge the result ###
  1 ### print(" Testing if we are looping/exiting the input section of the code properly")
  1 print(" This is crucial because if we exceed 1000 sequences it will be very time consuming. So the maximum allowable sequences limit is set to 1000 \n")
  1 print(" This programme will require 2 inputs from the User and will process and generate outputs depending on the input\n")
  1 print(" Time to generate a new dataset \n")
  1 print(" Type y if you wish to do so OR n if you don't want to/already know what this programme does\n")
  1 print(" We also cannot do alignment with just 1 or 0 sequences, so in these scenarios we need to generate new dataset \n")
  1 print(" We will be searcing for motifs in the PROSITE database using the sequences in the dataset\n")
  1 print(" Would you like to view what you will have to input and the list of outputs that will be generated?\n")
  1 print(" You have chosen not to proceed with the current dataset\n")
  1 print(" You have chosen to proceed with the current dataset\n")
  1 print(" Your current inputs were ", taxo_group, " & ", protein_sequence, "for reference\n")
  1 print(" Your current inputs were ", taxo_group, " & ", protein_sequence, "for reference\n")
  1 ### -v -v => command-line flags (explicitly and implicitly set) are printed in addition to the progress report
```

# Checking use of comments

```
70
  1 ### -auto => Turns off prompts
  1 # Calling a function asking the user if they want to proceed ###
  1 # Combining all patmatmotifs output files into one
  1 # Create a file for each element of the list i.e., for each sequence
  1 ### create a new directory for the current dataset, move all the files into this new directory and change the current working directory to the new one ###
  1 # Create directories for the files being created
  1 # Create directory for searching motifs from PROSITE db
  1 ### END OF PROCESSING STAGES ###
  1 ### END OF THE PROGRAMME, GOODBYE ###
  1 fasta_dir = f"{protein_sequence}_{taxo_group}_individual_fasta_files"  # Corrected
  1 ### --full-iter => iteration will be done in full mode not mBed mode (more acurate)
  1 ### --full => we want full distance matrix evaluation
  1 ### -graph -png => to get the output as an image which can be viewed
  1 ### Import the necessary packages ###
  1 ### --iter=1 => number of iteration will be one so we can use kimura corrections
  1 # Move fasta file to new working dir
  1 ### Move into the processing stages ###
  1 ### moving all the files ###
  1 ### NOTE_00_Details: We need store the current working directory of the user as a variable to ensure the user can run the programme from any directory and so that we can
  1 ### NOTE_00_STATUS: CLOSED ###
  1 ### NOTE_01_Details: Greet the User, display a list of what this program can do and what kind of inputs and outputs are part of this programme ###
  1 ### NOTE_01_STATUS: CLOSED ###
  1 ### NOTE_02_Details: The function below will keep requesting for an input till the user inputs either y or n (either uppercase or lowercase) or yes or no and display the relev
  1 ### NOTE_02_STATUS: CLOSED ###
  1 ### NOTE_03_Details: Now we will take the inputs from the user and run functions to see if they are in the right format for each input  ###
  1 ### NOTE_03_STATUS: CLOSED ###
  1 ### NOTE_04_Details: Now we will accept the input from the user for the taxonomic group that they want to search in and display some information regarding the closest ma
  1 ### NOTE_04_STATUS: CLOSED ###
  1 ### NOTE_05_Details: Now we will ask the user if they want to proceed to the next input i.e. the protein to be searched for ###
  1 ### NOTE_05_STATUS: CLOSED ###
  1 ### NOTE_06_Details: Now we will accept the input from the user for the protein that they want to search for ###
  1 ### NOTE_06_STATUS: CLOSED ###
  1 ### NOTE_07_Details: Now we will take the inputs and check the number of sequences that will be returned when we do the esearch command, we will also keep a limit on th
  1 ### NOTE_07_STATUS: CLOSED ###
  1 ### NOTE_08_Details: Now we will take the inputs and generate a file using esearch, efetch and xtract to get a ordered list of species sorted in descending order of their freq
  1 ### NOTE_08_STATUS: CLOSED ###
  1 ### NOTE_09_Details: Generate the fasta file which we will need for the further processing stages. Futher, asking the user if they are satisfied with the dataset that was displ
  1 ### NOTE_09_STATUS: CLOSED ###
  1 ### NOTE_10_Details: Ask the user if they are satisfied with the current dataset, give them option to generate a new one ###
  1 ### NOTE_10_STATUS: CLOSED ###
  1 ### NOTE_11_Details: Moving into the main processing stages. We will begin by first determining and plotting the level of conservation, we will be using clustalo to perform
  1 ### NOTE_11_STATUS: CLOSED ###
  1 ### NOTE_12_Details: Give the user the option to generate plots of the level of conservation using plotcon according to the winsize of their specification, put this in a loop to
  1 ### NOTE_12_STATUS: CLOSED ###
  1 ### NOTE_13_Details: Time to find if the dataset sequences have any motifs of interest that are present in the PROSITE database ###
  1 ### NOTE_13_STATUS: CLOSED ###
  1 ### NOTE_14_Details: We will now run a programme from the EMBOSS suite which has some biological relevance to our dataset. The name of the programme is backtranseq
  1 ### NOTE_14_STATUS: CLOSED ###
  1 ### --outfmt=fa => output will be in fasta format
  1 print("\n ### INPUT SECTION OF THE CODE ### \n")
  1 print("\n ### PROCESSING SECTION OF THE CODE: Level of conservation between the protein sequences ### \n")
  1 print("\n ### PROCESSING SECTION OF THE CODE: Scanning for Motifs in the protein sequences ### \n")
  1 ### print("\n ",protein_sequence," \n")  ### Used to gauge the result ###
  1 ### print(" ",taxo_group) ### Used to gauge the result ###
  1 ### print(" Testing if we are looping/exiting the input section of the code properly")
  1 # Running patmatmotif on each fasta file
  1 # Run the command using subprocess
  1 ### Script for ICA2: A Python3 workflow ###
  1 # Search, count, and list the motifs found from the sequences in descending order of frequency
  1 ### special parameters used
  1 ### special parameters used for plotcon
```

```
1 # Split the content into separate sequences and store as a list
1 # Store the content of the input fasta file into a variable
1 ### This function enables the user to generate as many plots with different winsizes for the current dataset as desired for maybe comparison with another dataset maybe ##
1 ### This function will take the user input winsize and use it's value in the -winsize parameter of plotcon ###
1 ### --threads=100 => to run the process on 100 threads
1 ### --use-kimura => applying kimura corrections on the protein sequences for a more accurate representation of evolutionary distance
1 #!/usr/bin/python3
1 ### -v -v => command-line flags (explicitly and implicitly set) are printed in addition to the progress report
1 ### -winsize 'winsize_input_user' => gives the user the option to choose winsize
```

All "legal" uses, thanks. Did you find the NOTE thing useful? You are the first person I have seen using it...!

- **Were any parts of the process multi-threaded?**
  `blast` wasn't used
  `clustalo` was explicitly (but defaults to all available threads anyway).
- **Did it use BioPython?**
  No, good!
- **Did it use Python modules (e.g. to interact with the OS)?**
  Yes, quite a few
- **Did it use Python pandas?**
  No, but there were a couple of places where it could have been useful!!
- **Did it use Python plotting (as opposed to EMBOSS plotting)?**
  Not really, but used image out of it

CODE

- **Was the user told what they can and can't do, preferably at or before the time they are doing it, by the code interface (rather than buried in the manual somewhere!?)**
  Yes, in great detail. Well done (but do lose some of the pauses, perhaps?!)
- **esearch/efetch: is the search specific enough**

```
egrep "esearch|efetch|xtract|efilter" *.py
```

```
esearch_command= (f"esearch -db taxonomy -query '{t_g}' | efetch -format xml |"
f"xtract -pattern Taxon -sep '|' -element TaxId ScientificName GenbankCommonName Division > '{t_g}.txt'"
os.system(esearch_command)
### NOTE_07_Details: Now we will take the inputs and check the number of sequences that will be returned when we do the esearch com
total_sequence_command  = (f"esearch -db protein -query '{protein_sequence}[Protein] AND {taxo_group}[Organism] NOT PARTIAL' |"
f"xtract -pattern Count -element Count"
### NOTE_08_Details: Now we will take the inputs and generate a file using esearch, efetch and xtract to get a ordered list of species sor
esearch_command_list =(f"esearch -db protein -query '{protein_sequence}[Protein] AND {taxo_group}[Organism] NOT PARTIAL' |"
f"efetch -format docsum -start 1 -stop '{total_value}' |"
f"xtract -pattern DocumentSummary -element Organism | sort | uniq -c | sort -nr > '{protein_sequence}_{taxo_group}.txt'"
subprocess.run(esearch_command_list, shell=True, capture_output=True, check=True)
gen_fasta_file =(f"esearch -db protein -query '{protein_sequence}[Protein] AND {taxo_group}[Organism] NOT PARTIAL' |"
f"efetch -format fasta -start 1 -stop '{total_value}' > '{protein_sequence}_{taxo_group}.fasta'"
print(f"\n Motifs present have been successfully extracted and saved to '{output_file_name}' \n")
```

  Yes
- **Correct: does the programme produce the correct output when run?**
  Yes, I think so
- **Robust to error: does the programme check for valid inputs/outputs while processing, and if not good, fail gracefully with a useful error message?**
  Yes.... but I broke it....sorry!
- **Well-structured: is your code divided into logical functions/units, where that is possible?**
  Yes
- **Well-documented: do the comments help a programmer to understand/maintain your code?**
  Yes, very well documented
- **Concise, well formatted: is your code laid out in a clear, consistent way?**
  Yes

# Were the commands correct?

- *** initial processing stages**
  Ideally one wants to take advantage of all the `edirect` functionality, forming commands inside functions such as

```
esearch_query="esearch -db protein -query \"{}[orgn] AND {}[Protein Name] NOT Partial\" | esummary|\
  xtract -pattern DocumentSummary -element AccessionVersion,Title,Organism,Slen".format(taxon,protein)
```

  and then, perhaps, read the fasta file into a dictionary, with the accession (or perhaps full header?) as the keys.
  The dictionary could then also be used for piping sequences as STDIN into motif searching, etc....
  Alternatively, this information could be brought into a pandas dataframe and then used throughout for the rest of the processing.
  Yes, you mostly did this, or something similar.
- *** final processing stages**
  `patmatmotifs` and wildcard done; not sure of the summary of outputs?
  The files in the patmatmotifs (and the individual files) directory didnt have recogniseable names, e.g. the accession ID would have been useful!?
- **Was there any two-way interaction with the user?**
  Yes, lots, almost too much even perhaps?!
- **If so, were there error traps for input?**

Yes, quite a few, often with similar function names??

- **If so, was all the input requested at the beginning, or did the user have to enter info at different stages as the programme was running?**
  Staged, which isn't an issue with the small dataset, but for larger ones, it meant I had to be there to wait for it.
- **Was the user ever given the option to continue or not continue?**
  Yes
- **Did the user have to do anything other than enter the taxonomic group and protein family?**
  Yes, make choices
- **Did the programme check how many species were in the requested dataset?**
  Yes, but I am not sure we got told how many there were, even though there was a listing of decreasing frequencies...
- **Did the programme tell us what the species were?**
  Yes
- **Did the programme check to see the size range of the requested dataset?**
  No
- **Did the programme let us filter on size?**
  No
- **Were there any progress indicators?**
  Yes
- **This code was first tried with Aves glucose-6-phosphatase. How did it cope with the bigger dataset (Eukaryota kinase)?**
  It wouldn't let me do them all as there were more than 1000 seqs.
- **Was an alignment done irrespective of the number of sequences in the initial set?**
  Yes, but the user was told if there were a lot of sequences, but there was an option to bail out.
- **Was the number of sequences reduced?**
  Yes, limited to the first 1000 I think?
- **Was the `cons` programme used, and what parameters were chosen to generate it (e.g -plurality, -identity, -setcase)?**

```
grep -w cons *.py
```

  No
- **Were all the outputs kept?**
  Yes, and all put in sensible folders. =-) THANK YOU!

## What were the analyses offered?

- **Conservation:**
  `plotcon` was used to produce the plot, with user able to modify the window size
  No explanation of `plotcon` output? What ARE the values on the axes?
  Graph not labelled for these data; underlying data were not assessed at all.
- **Motif search:**
  `patmatmotifs` was identified as the correct EMBOSS programme to use, with default parameters only, for all sequences.
  there are many different outputs possible, for example `-rformat excel` makes a nice report...
  Report files would have been more useful if the accession IDs were in the file names, so I would know which ones I might want to open.
  Outputs were partially assessed (but not summarised at sequence level (e.g. sequences with more than one motif)): outputs could have been made into a more detailed "catalog" of what was found (seq, motif, position in protein)?
- **wildcard option(s):**
  `backtranseq` was done, but there was no processing of the wildcard outputs that might relate them back to the conservation plot? What codon usage table was used, did it know to use the right one for the incoming dataset...? Looks like your code is running it for human sequences...?!

```
grep backtranseq *.py
### NOTE_14_Details: We will now run a programme from the EMBOSS suite
  which has some biological relevance to our dataset. The name of the programme
  is backtranseq. It reads a protein sequence and writes the nucleic acid sequence
  it is most likely to have come from. ###
print("\n When we run backtranseq on our dataset we get nucleic acid sequences that
the proteins in our sequences most likely came from \n")
backtranseq_command=(f"backtranseq -sequence '{protein_sequence}_{taxo_group}.fasta'
  -cfile Ehuman.cut -outfile '{protein_sequence}_{taxo_group}_nucleotide.txt'")
os.system(backtranseq_command)
```

## Did the programme meet all the requirements of the assignment

- **to identify a family of protein sequences from a user-defined subset of the taxonomic tree that could then be processed using, for example, one or more of the EMBOSS programmes installed on the MSc server**
  yes
- **to determine, and plot, the level of sequence conservation**

to show the plot, could have used `display` or `eog` or `xdg-open` or `firefox` or `image` from matplotlib for the file that was saved, or just use the `X11` option from `plotcon`...
plotted and saved, but output data not processed much further/summarised

- **to scan protein sequence(s) with motifs from the PROSITE database, to determine whether any known motifs (domains) are associated with this subset of sequences**
  Yes, but output data were not really summarised in a way that the user could identify which ones to look at in more detail
- **to do any other appropriate EMBOSS (or other) analysis that you think might add relevant biological information to the outputs**
  Yes, but wasn't quite sure what the biological relevance was, sorry...

## The user "experience"

- **Utility: does your programme present a seemingly useful tool for biologists?**
  Yes, with some tweaks as noted above
- **Usability: how easy is your programme to use?**
  Easy
- **Interface: does your programme present a consistent interface to the user?**
  Yes, nice interaction (but as noted above, too many pauses for me...)

## The user manual, which contains instructions for running the program, and interpreting the output

Write a "help manual" for your programme, which has two main sections;

- a section aimed at an "ordinary user": it contains instructions for running the program, and interpreting the output; it is aimed more at biologists who may not know much about Python3 coding, they just want the outputs!

  The user section of manual **doesn't** need to be huge. Remember that this is for a non-programmer to read, so you don't have to go into details of what variables you're using, *etc.*. It just has to describe briefly how the programme works, how to use it, and, perhaps, how NOT to use it!?

  You shouldn't need more than 5 pages of text, and probably fewer would be fine. Pictures are good here. You should use the outcomes from the test set as example inputs and outputs in this manual.
- a section aimed at a competent Python3 code-writer. This is usually called a "maintenance manual", which explains/shows how the different parts of the programme fit together. Feel free to use words like variable, function, iteration, error trap etc!

  This bit of the help manual should essentially be a description of the programme components and how they link together, **not** a listing of variables/functions/dictionaries/lists etc. Flow diagrams work very well here.

  You shouldn't need more than 5 pages of text, and probably fewer would be fine. Pictures are good here too.

Good to see a summary overview at the beginning so the user can judge whether they want to use the software in the first place!

- **Readability of documentation: is the biologist user manual section easy to read and understand?**
  Yes, the user is led through!
- **General description: is it easy for the user to understand how to run the programme?**
  Yes
- **indicates any things the user will have to do to make things work**
  Yes, clear instructions.
- **Is there test set example output shown?**
  Yes, from nearly all stages (`backtranseq` outputs not shown)

## The maintenance manual section, which explains how the different parts of the programme fit together

It is aimed at a programming-literate audience (so you should feel free to use words like variable, function, iteration, error trap etc); this should essentially be mostly a description of the workflow components and how they link together, rather than a listing of variables/functions/dictionaries/lists etc.

- **Readability of documentation: is the maintenance manual easy to read and understand?**
  Yes, just refer to the flow charts sooner!
- **General description: is it easy for a programmer to understand how the programme works?**
  Yes. Minor quibbles about font size on the flow charts (too much empty space, could have made boxes bigger?)
- **highlights any difficulties, if any, that you have come across**
  Not asked for, and none mentioned, liked the inclusion of "future modifications" sections. There is always room for these!

## Any final comments

Excellent effort, code and manuals. Do please note the comments above, they have been made in a constructive manner!
I hope you find at least some of this feedback helpful and constructive!
Cheers,

al

## Mark awarded: 85