**2025/05/18 (start of the project)**

What I’ve worked on:

* Learning Git
* Brainstorming about my project + trying to find suitable data in my field (biology), focused on genes, proteins and biological databases such as NCBI and UniProt.
  + Eventually setting on analysing conservation of protein sequences
* Reading up on many different Python libraries which I could possibly use on protein sequences to analyse conservation, including Biopython (very large library with many different modules and applications), FEPS (Feature Extraction from Protein Sequence), biotite (aligning amino acid sequences and generating phylogenetic trees), peptides (for acquiring peptide statistics, sequence profiles, and physiochemical properties, just from the amino acid sequence), and SeqVec (machine learning model for creating embeddings for amino acid sequences).
  + Eventually deciding on BioPython because of its extensive functionality and applications, aided by a module from peptides library
* Experimenting with importing and reading .fasta files in Python using the SeqIO module from Biopython library

Encountered problems and resources I used to solve them:

* Needed to do some additional steps which took some time to get git clone and git push working because authentication methods have been changed in git since 2021/08/13. A geekersdigest.com blog post was published which explained how to use tokens instead.
* The information on libraries on their respective documentation pages or developer GitHub repository pages, (such as biotite-python.org, github.com/dukkakc/FEPS\_CFS, pypi.org/project/peptides/, and pypi.org/project/seqvec/ for FEPS, biotite, peptides, and SecVec, respectively) shows great potential, but many are very complicated to learn and require very specific input data such as multiple sequence alignments (MSAs), protein structure data, or even data preprocessing in bash (for instance ProDry and Evol). Despite being useful skills to learn, they are not suitable for my 40-hour project, and I want to keep the problem simple with not too complicated data. Many libraries are too complicated to use as the main new library for my project, but on the other extreme, some tools are only available as complete Python scripts which should be run from the terminal directly, with no option to see and adapt the code to your needs, such as iFeature, which is also not useful for a project where the goal is to learn using a new library. After reading some of the Biopython documentation on biopython.org, it seemed perfect for this project, because it requires minimal data input and despite the library being extensive, you do not have to learn every single function of it.

What I learned:

* Learned to use Git
* Learned to use the SeqIO module from Biopython to import amino acid sequences
  + Currently downloaded proteins from three different organisms which fall within the GPCR family, but this family is very extensive and is an umbrella family for a lot of different receptors with lots of different functions and which also interact with many proteins and ligands. I am not sure whether choosing a protein with large variation within an organism will help me in comparing the proteins between organisms, so might change my choice for GPCR family proteins later.
* Learned a lot about the potential of different Python libraries in biology

**2025/05/19**

What I’ve worked on:

* Decided to switch from protein family to single protein to improve comparison between organisms and make the project feasible. Chose serine/threonine protein kinase B (Akt1). Downloaded verified DNA coding sequences .fasta sequences from NCBI (so not amino acid sequences as for GPCR) for six organisms: *D. melanogaster*, *D. rerio*, *G. gallus*, *M. mulatta*, *H. sapiens*, and *R. norvegicus*. Chose model organisms which are quite far apart (from fruit fly to fish, to chicken to mammal) to compare distant species. Could use the same code for importing and reading files which I already had for the GPCR files.
* Produced amino acid sequences from DNA sequences and stored them in .fasta files

Encountered problems and resources I used to solve them:

* I got a lot of errors and unexpected output from confusing Seq() and SeqRecord() objects. The Biopython documentation in addition to some posts on StackOverflow helped me understand the different syntaxes.
* I had only worked with nested dictionaries once before. Took most of my inspiration from the level\_9 assignment (because they are quite similar to .JSON files)
* During reading of the .fna files, I found out that some files contain multiple Akt1 DNA sequences with different accession numbers which SeqIO.read() cannot handle. I could use SeqIO.parse() for this, but to keep the upstream analysis feasible, I just removed the additional sequences, as they are largely the same and the purpose of the project is to learn the library, not analysing the data.
* I wanted to translate DNA to amino acid sequences and then align, but I required to write .fasta files for the amino acid sequences in between which was not covered clearly in the Biopython documentation. StackOverflow had a nice solution for this: using SeqIO.write()

What I learned:

* Handling .fasta sequences in Python
* Understanding the difference between Seq() and SeqRecord() objects and when to use which
* Translating DNA sequences to amino acid sequences using the .translate() attribute
* Writing .fasta files for DNA and amino acid sequences with appropriate annotations
* Handling nested dictionaries

**2025/05/20**

What I’ve worked on:

* Generating multiple sequence alignments (MSAs) and phylogenetic trees for Akt1 DNA and amino acid sequences
* Making graphs for amino acid composition and compare physiochemical properties between amino acid sequences using peptide library

Encountered problems and resources I used to solve them:

* I wanted to run an alignment, but this is only possible using a command-line tool. I solved the problem through using the subprocess library as suggested by the Biopython alignment documentation, allowing Python pipe a ClustalW command to the command-line.
* I felt like I needed at least a basic understanding of the ClustalW alignment tool before using it. To understand better how ClustalW works and how it should be used, I called the help in bash using clustalw -help, read a short documentation on portal.supercomputing.wales, and a slightly more elaborate documentation on vcru.wisc.edu.
* The standard output prints only the first fragment of the alignments which is arranged in columns. On biostars.org someone defined a slidingWindow() function and called it “awkward”, asking for input for a better solution, but I actually found it quite smart and took inspiration from it for my show\_alignment() function.
* I had no clue how to make a stacked bar plot. Apparently, you can first put your data in a pd.DataFrame() and then use ax = df.plot.bar(). This I actually found in the Pandas documentation.

What I learned:

* Using the subprocess library to pipe commands from Python to the command-line
* Writing a ClustalW command and understanding and interpreting the output
* Using the Phylo module to draw phylogenetic trees
* Make a stacked barplot using Pandas
* Acquire physiochemical properties from amino acid sequences using peptides library

**2025/05/21**

What I’ve worked on:

* Computing conservation values for amino acids
* Finding conserved motifs across the amino acid sequences using the Motif module
* Designing a weblogo for conserved sequences

Encountered problems and resources I used to solve them:

* No documentation available on how to interpret the matrix, so thought of a meaningful way to compute a conservation value myself.
* Substitution matrices are np.arrays but the row and column labels are apparently added separately, so using np.sort() only sort the values but not the corresponding labels. In the Numpy documentation, np.argsort() provided a solution, because I could apply it as index labels in a pd.DataFrame to order both the values and corresponding labels appropriately (we probably used np.argsort() before but could not remember)
* The Motif module was originally designed for DNA sequences only, but updating from version 1.81 to 1.85 changed the alphabet from fixed symbol sets such as alphabet=alphabet\_dna to allowing string input with any symbol, so I could use all amino acid one letter codes. I found this out through reading the motif documentation for different Biopython version which is updated quite a lot.
* The motif.weblogo function is broken and returns a .png file which cannot be opened. It has been a persistent problem which dates back years according to the many forum posts I have read. I eventually used the weblogo.berkeley.edu graphic interface which I reached from an external link in the Biopython documentation.

What I learned:

* Making and interpreting substitution matrices
* First time using np.apply\_along\_axis()
* Using the Motif module for computing consensus sequences
* Designing a conservation logo

**2025/05/22**

What I’ve worked on:

* Matching full amino acid sequences to a domain database (not without problems, read below!)
* Acquire matched domain consensus sequences from database and storing them in a dictionary.

Encountered problems and resources I used to solve them:

* It was actually really hard to find a database of protein domain sequences (rather than full proteins). In addition, the databases which had domains either had domains available for specific species proteins only (Johnson lab libraries), were multi-species but unavailable for download (MEME Suite and PBD), or could only be blasted using Graphic Interface (ELM). Eventually found Pfam which is multi-species can be run from the command-line and thus from Python through subprocess. I read the documentation for using hmmer to access the Pfam database on hmmer.org
* I wanted to avoid downloading a database file every time I ran cell 15. I found the answer on StackOverflow: using Path from pathlib to check whether a file exists in a directory before downloading it.
* Initially my pfam\_results.txt file was not imported in pandas correctly due to varying spaces between columns. Geeks for geeks had a solution, using "\s+" as separator
* On StackOverflow I found how to use a lambda function to fuse columns, because I had to fuse the last few columns to correctly read the description of target names
* I wanted to acquire the found domain sequences from the Pfam database to compare them to my consensus sequences. I found a useful function called fetchPfamMSA () in the Prody library which documentation can be found on bahargroup.org. Unfortunately, this library uses a way outdated version of Numpy and despite trying a lot of different things, I could not get it to work. I eventually had to retrieve the sequences manually from NCBI, instead of from Pfam using the domain names. You cannot find the sequences on the Pfam website, and opening the database in a text editor uses a specific format with coordinates suitable for hmmer but not for directly accessing sequence either. Too bad, I tried my best but cannot fully automate it now and also not use the same sequences which were matched.
* Because the consensus sequences of the domains contained many gaps and were not available for downloading, I had to manually make a .fasta file in a text editor. Two problems were that 1) .fasta sequences never contain gaps, so I had to put in the individual fragments with unique ids using \_1, \_2, etc. and 2) text files contain hidden format, which I solved by importing the sequences, cleaning them, then resaving them as .fasta files using SeqIO.write(). I deleted this part of the code because it is not relevant to the project subject, but the resulting file is saved as ./data/consensus\_matched\_domains.fasta. This would have been the result of fetchPfamMSA().

What I learned:

* It can be a hassle to find a database for your specific needs.
* Always read the documentation of a library well before installing to make sure it is compatible with your versions of the libraries which it is dependent on.
* Dealing with reading tables from divergent file formats

**2025/05/23**

What I’ve worked on:

* Aligning conserved motif to domain sequences
* Blasting human Akt1 coding sequence to NCBI coding sequence database
* Drawing phylogenetic tree from blast output

Encountered problems and resources I used to solve them:

* Biopython Pairwise aligner accepts gaps in input sequences, but counts the score +1 if a gap aligns with an artificial gap in the sequence to be aligned introduced by the aligner itself. For instance if AA-CC is aligned with AACCG the aligner may create a gap between AA and CC in the latter to optimise the alignment, but then counts the gap in the first sequence as +1 because it now aligns with a gap. This was not clear from the Biopython documentation. To avoid this, I replaced the gaps with letters that are not one letter amino acid codes (X and Z).
* I first thought you could make a phylogenetic tree from a blast output XML file right away. Actually, in fact you need alignments in the XML file. Therefore, you need to run ClustalW again. I found how to extract the sequences from the blast XML file on StackOverflow. Then running ClustalW using subprocess was easy because I had done it many times.
* ClustalW truncates ids after 30 characters, but I wanted to show the species names which sometimes exceed this, resulting in duplicate names and errors. Found solution on biostars.org -> use enumerate() then put the index in front of duplicate names to provide unique ids anyway

What I learned:

* Using the Biopython pairwise aligner
* Running non-graphic interface BLAST (have done only graphic interface ones in the past)
* Dealing with and avoiding duplicate identifiers in ClustalW
* Drawing phylogenetic trees using Phylo.draw() from .xml files rather than .dnd files which requires an additional distance calculation step.