# Programming Project 05 - Cross Verification of Protein Information

## Group Projects

These projects serve as a way for you to learn how to build a programmatic tool in a group setting. Often in this field, large packages and software are built in a collaborative effort, and knowing how to effectively communicate ideas, tasks, and workflow is an essential skill. Here, you will work as part of a group and demonstrate your ability to compose a cohesive (and working) tool comprised of components created by others *and* yourself.

## Deadline

**February 08, 2021 - 12:00/noon (UTC+1:00)**

## Submission Guidelines

1. Clone your repository to your local workstation/computer if you have not done so
2. Structure and work on your submission.
3. Commit your work to the git repository.
4. Create a git tag.
    - Tag name must be equivalent to ”GroupProject”.
    - Tag can be made via the command line or using the GitLab GUI
5. Be sure to include a PDF of your presentation in your repository once it is finished

## Package Requirements

- All code comprising the backend portion (i.e. the code responsible for downloading, parsing, and formatting the information) must be compiled as an installable Python package
- The package must contain the following:
    - A working CLI (see CLI Requirements below)
    - A clear and descriptive `README.md` that details what the package is, what it does, how it can be used, and examples of how to use the CLI
    - The necessary dependencies so that the package works immediately upon installation in a new virtual environment
    - Working unit tests that test at least 70% of the code in the package

## CLI Requirements

- Within the python package described in the package requirements section, there must also be a working command line interface (CLI)
- CLI methods must contain proper documentation that is accessible via the command line
- CLI method documentation should contain:
    - Explanations of the arguments and options involved with the method
    - Brief description of what the method is used for

## Use of External Libraries

In general, one can make use of an external library or package that can aid in accomplishing a small subtask, such as a combinatorial problem, interface with an API, etc., but you cannot use a library or package capable of solving **all** of your tasks. You are of course allowed to use modified code from your previous individual assignments (including that of PLAB1) where

applicable. If you do choose to use an external resource to perform part of one of your tasks, it must be properly explained in the presentation. If you have any questions or concerns about whether a particular resource is allowed, please feel free to ask via email or issue.

## General Remarks

- The tasks are purposely written in such as manner as to require you, as a group, to figure out what tools are needed, what information needs to be gathered, and what resources should be used
- All code-based work is to be done in GitLab
- Use GitLab Issues to track and assign individual tasks and required work
- The software package (backend code) and web application (frontend) can be stored in separate folders in the root directory of your repository as shown here (you can rename these folders as you please):

```
├── frontend
└── project_package
```

## Grading (10 pts):

| Task | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Points | 4 | 1 | 2 | 3 |

# Cross Verification of Protein Information - Introduction

In the field of bioinformatics, there are certain "go-to" databases that are assumed to be the "gold standard" for information pertaining to a particular topic. For example, UniProt is generally considered as the primary source of all information pertaining to proteins. They earned this title due to their 2-part system (Swiss-Prot and TrEMBL) responsible for extracting, identifying, and curating as much protein information and data possible. Many other databases cross-reference these "gold standard databases" on their own site, and scientists often trust that if the information is referenced, then it must be the most up-to-date information available. However, we all know that saying about assumptions…

In this project, you will investigate whether a particular set of protein information available on two different "gold standard databases" match up. You will extract relevant and comparable information from these databases for a given query and check whether everything is properly updated.

# Aims

1. **Download and parse** human protein information from UniProt and NCBI's RefSeq
2. **Map comparable metadata** from the databases to one another
3. **Calculate** what percentage of the databases' information aligns
4. **Create a frontend** which allows one to see the overall comparison of the databases as well as check how well the information aligns between the two databases for a given protein

# Tasks

## Task 1 - *Database Download* (4 pts)

- Download and compile data from UniProt and RefSeq pertaining to *all human* proteins so that you have the following information for each protein from both databases:
  - Symbol (i.e. gene symbol)
  - UniProt accession identifier
  - RefSeq accession identifier
  - Amino acid sequence
- *Hint* - Though the user of your package may need to download a large file at first, be sure your software parses these files for the relevant information and caches it in smaller files elsewhere for faster lookup the next time it is run

## Task 2 - *Mapping the Metadata* (1 pt)

- Map the metadata extracted in Task 1 to their corresponding partner. For example, a protein's gene symbol may be called something different in UniProt vs. RefSeq so make sure you are mapping comparable values from each database to one another

## Task 3 - *Compare the Data* (2 pts)

1. Create methods to compare a protein's metadata across databases. These methods should check the following:
   - Is the protein's gene symbol the same between the databases?
     - A gene symbol (also known as a gene name) is a unique combination of letters and numbers used to describe a DNA segment that contributes to phenotype/function
     - This differs from the HGNC ID which is also unique but comprised only of numbers
   - Do the UniProt accession IDs match?
     UniProt accession numbers typically appear as such: P29358, Q04683, etc
   - Do the RefSeq accession IDs match?
     - RefSeq accession numbers will appear as such: NP_000408, NM_000014, etc
     - RefSeq accession numbers can be followed by a period to indicate a version e.g. NP_000408.1

    - Are the amino acid sequences for the given protein the same length?
    - Do the amino acid sequences for the given protein align?
2. Add a feature to your software which summarizes this information for all human proteins i.e. gives a percentage of how many proteins' data match between databases for each metadata category listed above

## Task 4 - *GUI* (3 pts)

- Construct a web interface that displays the summary information all of human protein metadata tabulated in Task 3 as well as brings up specific information for a given UniProt accession identifier, RefSeq accession identifier, or gene symbol. Your interface should include the following features:
    - A table displaying the summarized protein comparison calculated in Task 3, part 2
    - A text box + submit button for inputting a UniProt accession identifier, RefSeq accession identifier, or gene symbol
    - A table displaying information on the comparison of database information (detailed in Task 3, part 1) for the given protein

## Hints

- Accession numbers (both from UniProt and RefSeq) can have *secondary* numbers which are those that were replaced with a new *primary* accession number in order to avoid redundancy. It may be worth checking the secondary numbers if listed accession numbers don't match…