## Running the Del Sequencing Tool

This notebook serves as a replacement for the earlier sequencing data processing pipeline that was used previously for NADEL. Several improvements have been made, namely:

- An external web-based preprocessing step is no longer needed and has been directly integrated into the pipeline.
- The final output is a DataWarrior-ready csv file featuring either absolute or normalized sequence counts. No further manipulation in Excel is needed.
- The prior sequence searching algorithm did not attempt to correct error sequences. Some error sequences are able to be recovered using this processing method.
- The code in this workflow is ~500 lines shorter than the previous program and is annotated more clearly. It is also designed to be more modular and making modifications and updates to suit any library is considerably easier to manage.

### Introduction –

This Del Screening Tool exists as a Jupyter Notebook and features several code cells that have been logically separated according to function. These cells are executed sequentially to move data through the full processing pipeline.

*What does it do?*

The tool takes a .fastq file and ultimately returns two .csv files containing normalized and absolute sequence counts organized according to BBs and screening conditions.

*What Libraries are supported?*

Currently, libraries with up to 4 encoding regions are supported. This translates to a maximum of 2 BB codes and 2 PCR codes. The code can be easily modified to support more codes should the need arise in the future.

### Package Contents –

| Name | Date modified | Type | Size |
|---|---|---|---|
| classes | 3/13/2024 2:28 AM | File folder | |
| docs | 3/4/2024 12:08 PM | File folder | |
| library | 3/12/2024 10:18 PM | File folder | |
| outputs | 3/12/2024 10:19 PM | File folder | |
| utils | 2/28/2024 2:40 PM | File folder | |
| DelScreenProcessing.ipynb | 3/14/2024 6:40 PM | Jupyter Source File | 12 KB |
| ExampleData.fastq | 3/4/2024 12:53 PM | FASTQ File | 71,173 KB |

- ***DelScreenProcessing.ipynb*** – The main notebook file used for data processing.
- **classes** – A subdirectory used as a python package for holding class modules.
- **library** – A subdirectory used as a python package for loading library-specific information.
    - ***primers_all.py*** – A file for storing a full list of PCR encoding sequences and IDs.

- ***primers_screen.py*** – A file for storing only the PCR encoding sequences and IDs used in a specific set of screens.
- ***fragment_codes.py*** – A file for storing the encoding sequences and id's for the building blocks in each diversity element.
- ***fragment_smiles.py*** – A file for storing the smiles and id's for the building blocks in each diversity element.

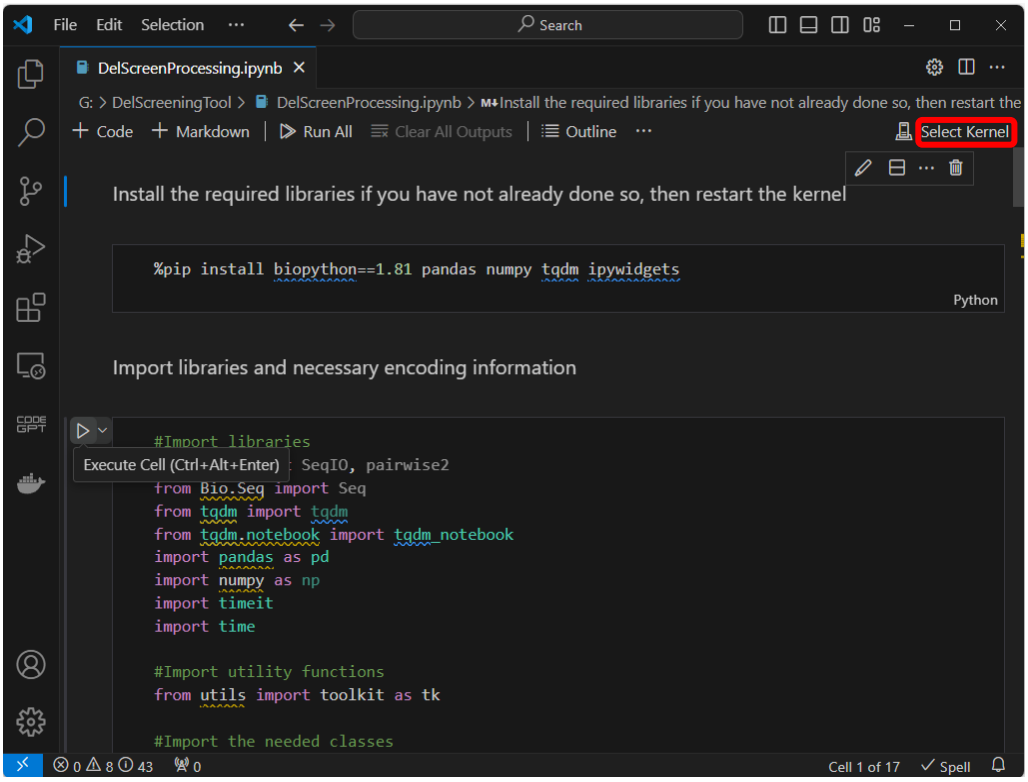
- **utils** – A subdirectory for additional utilities.
  - ***DictFormatter.ipynb*** – A notebook for auto formatting entries that can be pasted into the library .py files. Read more about this in the customization documentation.
  - ***SeqView.ipynb*** – A notebook for examining preprocessed fastq files. Useful for determining the slice indices for the encoding sequences.
  - ***toolkit.py*** – A file containing a variety of utility functions used by the main notebook.
  - ***ExampleCodes.csv*** – A csv file used as an example input for the DictFormatter notebook.
  - ***ExampleData_Preprocessed.txt*** – A txt file used as an example input for the SeqView notebook.
- **docs** – Documentation for the setup and usage of the included files.
- **outputs** – Files generated by the main notebook are organized and output here.
- ***ExampleData.fastq*** – A small .fastq file containing 200,000 NADEL reads that can be used for testing.

These files must be kept together in the same folder. The .ipynb file calls on the other files for functionality.

**<u>Using the Notebook with VS Code –</u>**

This tutorial assumes the use of sequences coming from the NADEL Library. If you need to use another library, see the **Customization.pdf**

Use VS Code to open **DelScreenProcessing.ipynb**



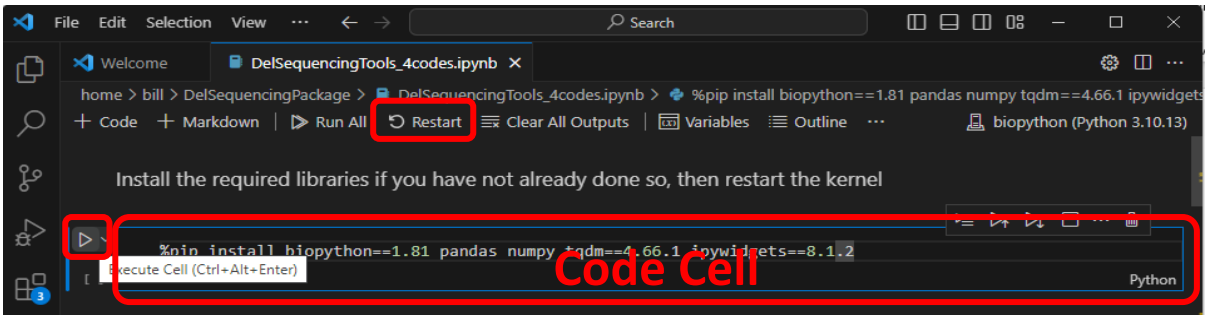
Now tell VS Code to use the biopython venv we created earlier to run this notebook (above screenshot).

In the upper right, click: Select Kernel > Python Environments… > biopython (Python 3.10.8)

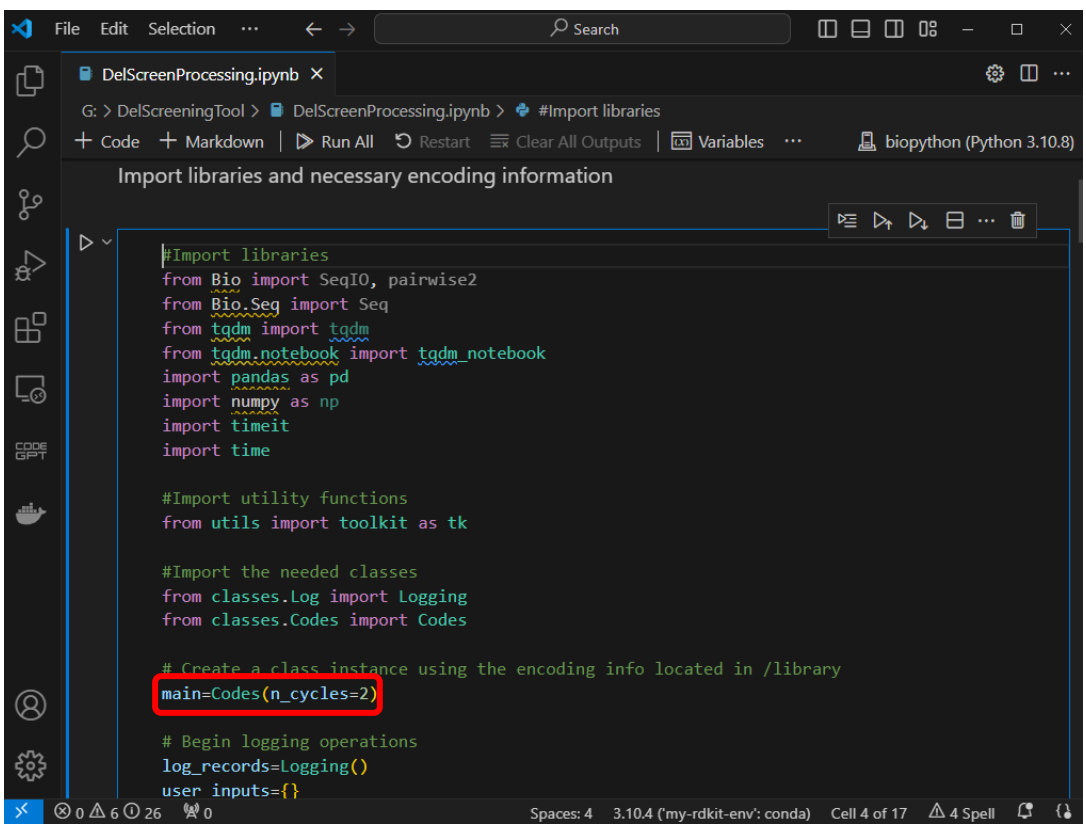
Notebook cells are run by mousing over them and then clicking the “Play” icon that appears just to the left of the cell.

For the first time running this notebook, lets go ahead and run the top cell to make sure all the necessary libraries are installed and up to date. After doing this restart the kernel. See the screenshot below for reference.

You should only need to run this topmost cell once.




Next, we want to import the python libraries we are going to be actively calling upon with the rest of code cells (below). We also will import the data contained in the other .py files and define a class (“main”, in this example) to hold our encoding sequence information.

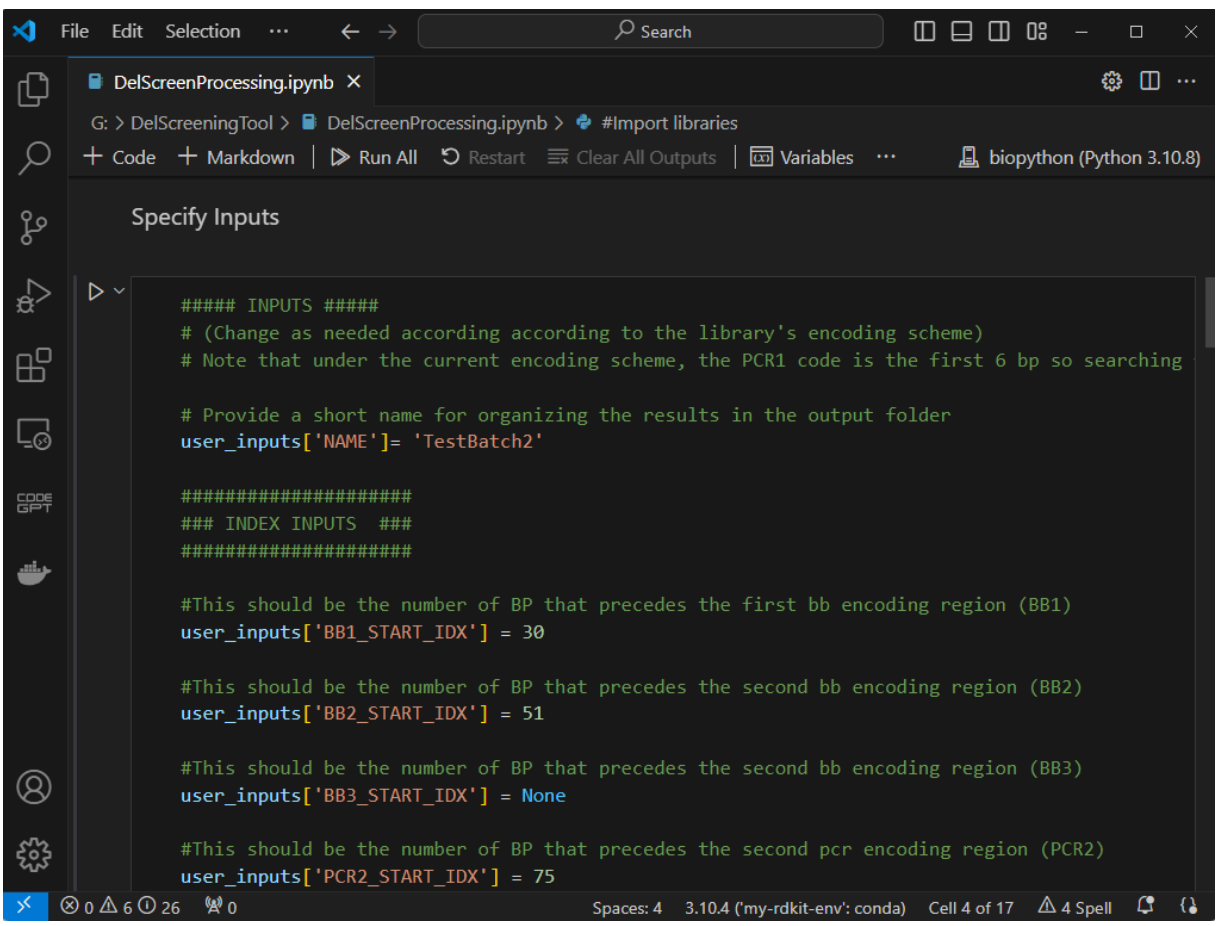


Be sure to set the “n_cycles” argument to the appropriate number of diversity elements in your library, then run the cell.

Before running the next cell which accepts all the rest of the user inputs (below), familiarize yourself with the different variables in this section. These are already predefined for the NADEL library but you will need to change them accordingly if a different library is being used. For more information see the **Customization.pdf.**

For now, if you are using a fastq file other than the current one provided for testing (ExampleData.fastq), drag the fastq file you want to process into the same folder with the rest of the package files and update the RAW_FASTQ_FILE variable to match the name of your file. Use the NAME variable to enter a short identifier for the screen(s) associated with your raw fastq data (this will be the name of the subfolder used for outputting results. For libraries other than NADEL, you will have to determine the BB and PCR index values.

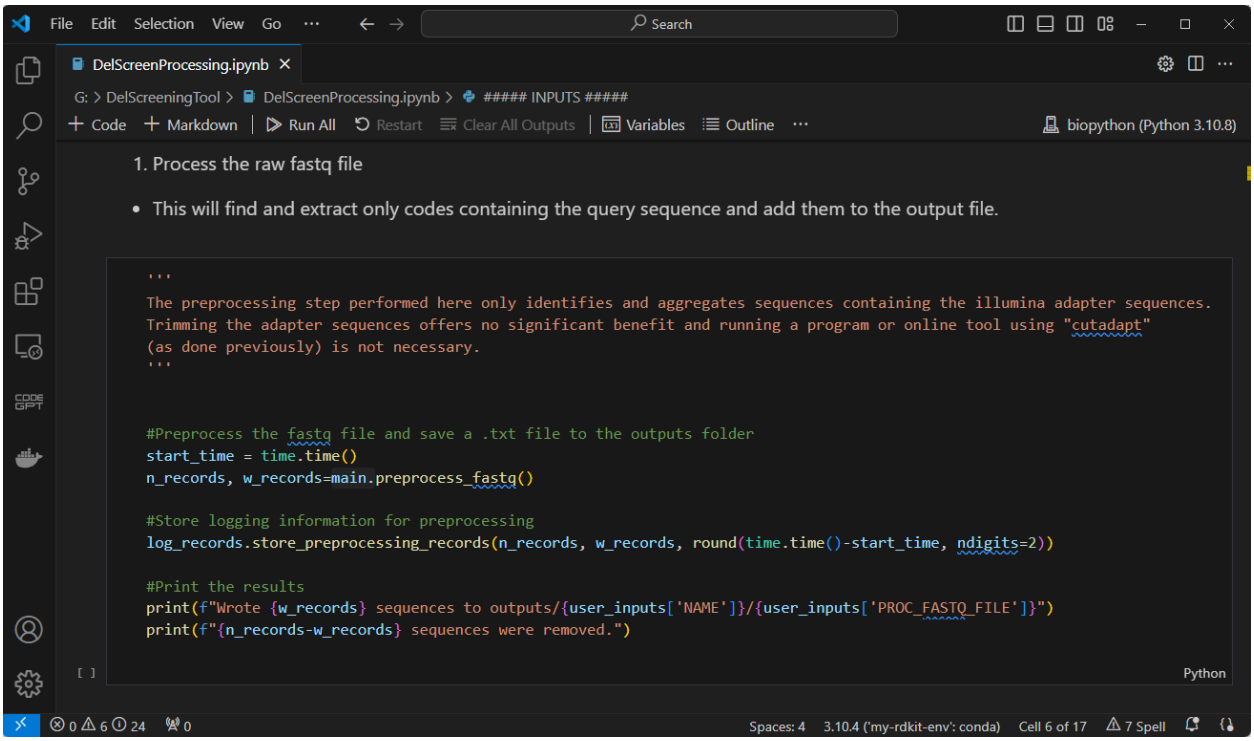
After making any other desired changes, run the cell.




We are now ready to begin our Data processing process.

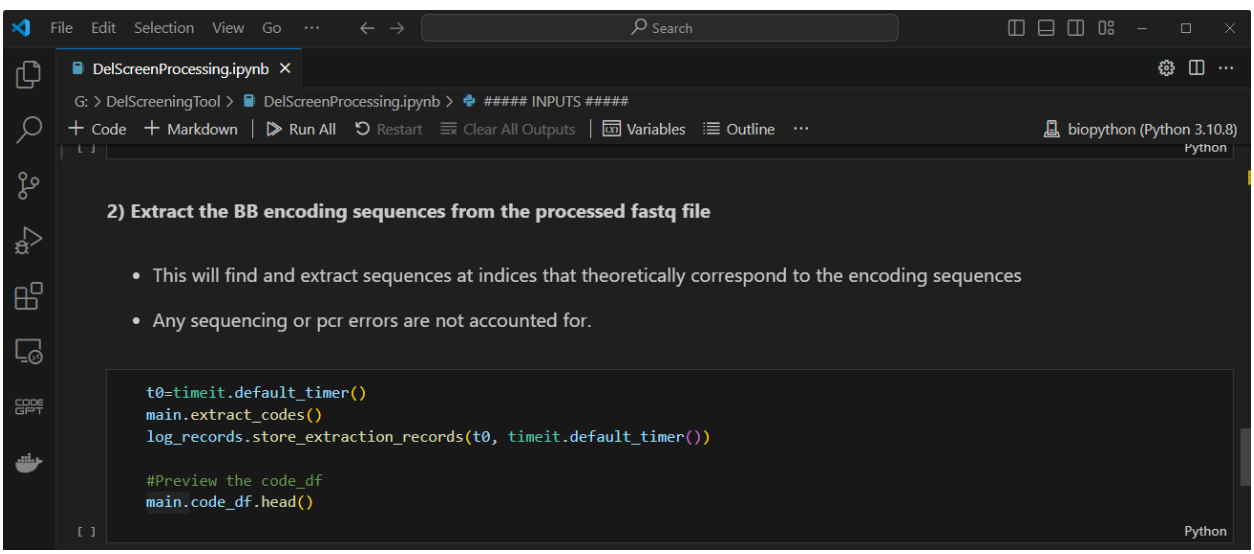
First let’s run the next cell (below) to extract only sequence reads that contain an Illumina primer sequence and add the sequence information to a separate txt file (created automatically in the outputs

folder). For our purposes, the fastq IDs and qualities are removed since they only increase the file size and runtime.



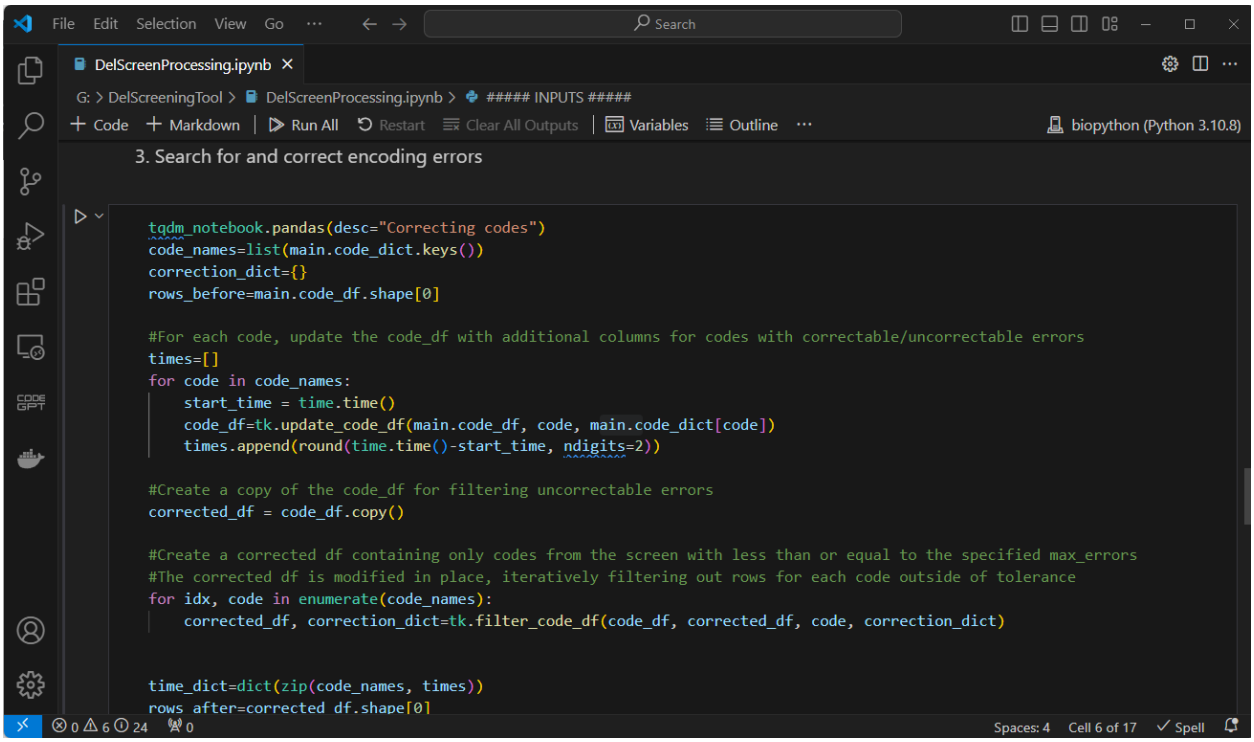

Next, lets extract the encoding information from the preprocessed sequence reads.

This method “slices” the encoding sequences out of the sequence read using their expected positions.

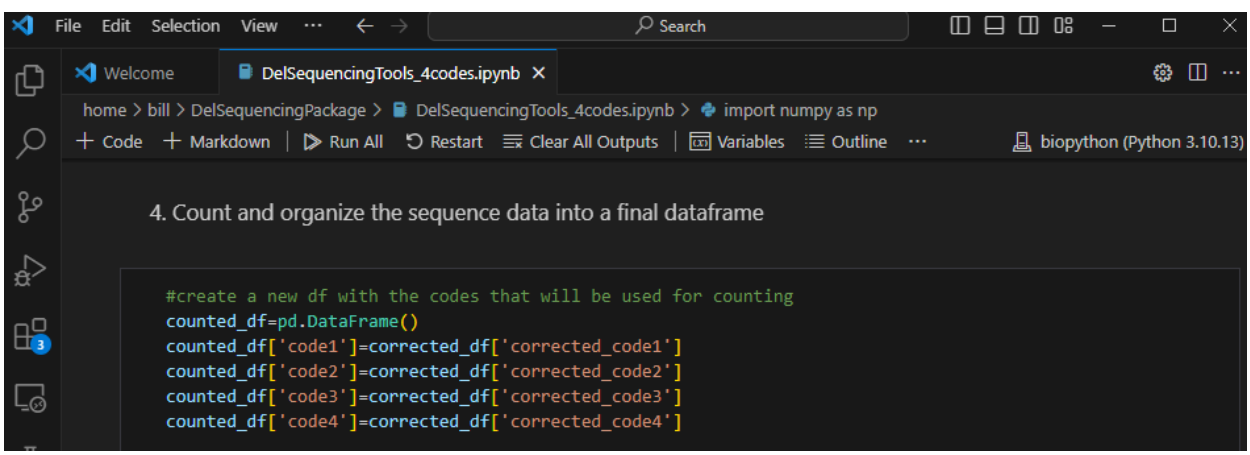


We next will apply a code correction step. Since our encoding sequences are unique from each other at 2 or more positions, if only a single error is present in the sequence, we can correct it in some cases. Reads with 2 or more errors are rejected and removed from the final counts.

Run the following cell (below) to perform the encoding corrections.




Next, we will combine and count identical reads into a final table which is done by running the penultimate cell (below).




Finally, run the last cell (below) to output your results as two csv files. One csv will have absolute sequence counts and the other will have normalized sequence counts.

DelScreenProcessing.ipynb

G: > DelScreeningTool > DelScreenProcessing.ipynb > ##### INPUTS #####

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline

biopython (Python 3.10.8)

5. Output the results in two csv files:
    - 1. As absolute sequence counts
    - 2. As normalized sequence counts

```python
# Merge column names
merged_columns = [''.join(col).strip() for col in final_df.columns.values]
merged_columns = [str(col).replace('1a-','') for col in merged_columns]
merged_columns = [str(col).replace('1b-','') for col in merged_columns]

# Assign merged column names to DataFrame
final_df.columns = merged_columns


final_df.to_csv(f'{main.output_location}/Results_AbsSeqCounts.csv')

#Convert to normalized sequence counts
#pcr_list=list(pcr_primer_1.values())
for pcr in final_df.columns[4:]:
    sum_value = final_df[pcr].sum()
    final_df[pcr]=final_df[pcr].apply(lambda count: (count/sum_value)*(len(final_df)))

final_df.to_csv(f'{main.output_location}/Results_NormSeqCounts.csv')
```

0 6 24 0 Spaces: 4 Cell 6 of 17 Spell