DOCUMENTATION FOR THE VARIANT CLASSIFIER SCRIPT:

Running the script:

The script that I wrote processes GFF, VCF and FASTA files and classifies the variants into three categories:

- Non-coding-
  These variants occur beyond protein coding regions and do not affect the protein sequence.

- Synonymous-
  These variants occur within a coding region, they affect the DNA sequence but have no effect on the translated protein.

- Non-synonymous
  These variants occur within a coding region but they alter the amino acid sequence, affecting the functioning of the protein.

The script also comes equipped with error logging, handling variant quality and generating output in a readable formatted way through a designated output file and a plot to visualise the distribution of the variants across the three categories.

Command to run the script:

```
python3 BIOL5381_3046323_2025.py --vcf variants.vcf --gff annotations.gff --fasta genome.fasta
```

The above command, when run in the terminal after adding the correct file paths, runs the script. The arguments are explain below:

- --vcf => path to the Variant Call Format (VCF) file. This file contains the variants and data on them.
- --gff => Path to the General Feature Format (GFF) file. This file contains data on the genome annotations.
- --fasta => Path to the FASTA file, standard file containing the chromosome id and the associated sequence.

In case any file is not found the script immediately logs an error in the log.error file and exits.

Exploring the Input File Formats:

Variant Call Format or VCF File:

This file contains data on the variations found, this includes SNP's, Insertions and Deletions. The columns we used from this file are as follows:

Chrom	Marks where the variant is found.
Pos	The exact location of the variant on the chromosome.
Ref	The nucleotide being referred to.
Alt	The alternative nucleotide.
Qual	Metric to decide quality of the variant

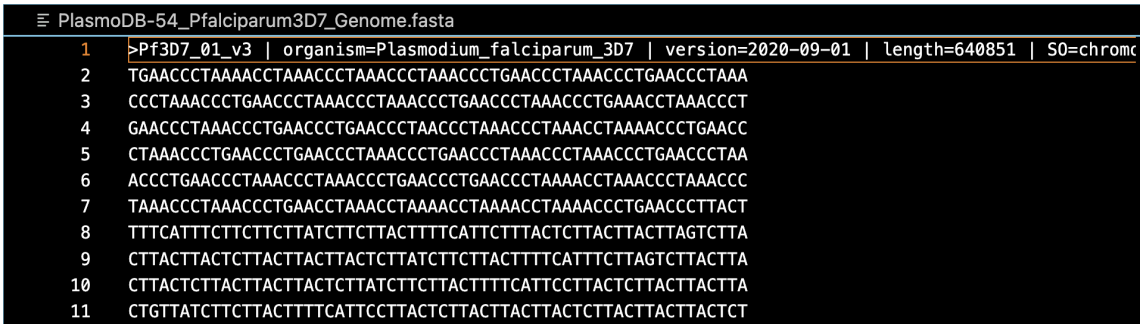
General Feature Format or VCF file:

This file contains data on where different things such as, the genes, exons, etc are found across the genome. The columns are as follows:

Sequence name	Identifies Chromosomes or other features
start	Marks the start of the feature
end	Marks the end of the feature
strand	Marks the direction to read the strand (+/-)
feature	Explains the annotation type
attributes	Metadata

Fasta File:

This file is a standard file that contains the full genome sequence with the associated sequence ID in the header. Here is a snippet of the Fasta format, from the VScode editor:



Script work-flow:

Processing and handling the files:

- The script ensures that the VCF file is read line by line, so that coordinates of all the variants are obtained.
- The script parses the GFF file and extracts all the Coding sequence regions (CDS).
- The script loads the Fasta file with function specificity so that it can retrieve the reference sequences.

Processing the Variants:

- The script uses a function, *coding_region_checker(),* to verify each variant on whether it exists within a coding region.
- The script then uses the function *amino_seq_checker(),* to translate the amino acid sequence of the verified variants.
- Once this has been done the script proceeds to classify the variants into “Non-coding” if the variant is outside the CDS, “Synonymous” if the amino acid remains unchanged, between the reference sequence and the mutated translated sequence, and “Non-Synonymous” if the amino acid is unchanged.

Logs and Outputs Data:

- The script comes equipped with logging capabilities allowing it to clearly inform users about the issue and the order in which they occur.
- The script skips variants that are of low quality i.e, a qual score less than 20.
- The output results are saved in a TSV file titled *Output_Variants.tsv.*

- A plot is generated to visualize the distribution of variants across categories, and is saved in the file *Variant_Numbers.png.*
- If an error is observed it gets logged in the *error_log.log,* the statistics are also sent to the same file.

Output Files:

Output_Variants.tsv:

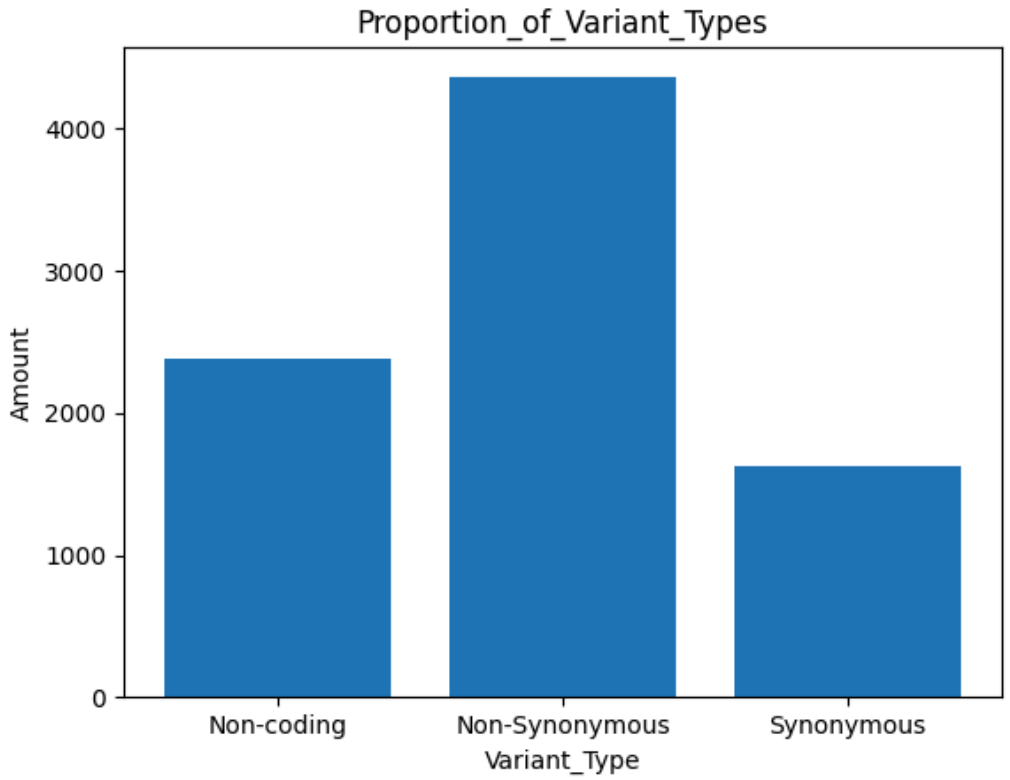
This file contains the data on the classified variants in the form of a tab separated table. Below is a snippet of the file from the VS code editor:

Output_Variants.tsv > data

	Chrom	Pos	Ref	Alt	Type	Transcript	Protein Location	Ref AA	Alt AA
2	Pf3D7_14_v3	52	G	A	Non-coding	NA	NA	NA	NA
3	Pf3D7_14_v3	663	C	A	Non-coding	NA	NA	NA	NA
4	Pf3D7_14_v3	2932	A	G	Non-Synonymous	PF3D7_1400100.1-p1-CDS1	514	S	G
5	Pf3D7_14_v3	4397	G	A	Non-Synonymous	PF3D7_1400100.1-p1-CDS2	120	S	N
6	Pf3D7_14_v3	4890	C	T	Synonymous	PF3D7_1400100.1-p1-CDS2	284	D	D
7	Pf3D7_14_v3	4935	G	A	Synonymous	PF3D7_1400100.1-p1-CDS2	299	L	L
8	Pf3D7_14_v3	4973	T	C	Non-Synonymous	PF3D7_1400100.1-p1-CDS2	312	I	T
9	Pf3D7_14_v3	5016	C	T	Synonymous	PF3D7_1400100.1-p1-CDS2	326	S	S

Variant_Numbers.png:

The plot that visualises the distribution of variants across the three categories. Below is the generated plot for the given data.



error_log.log:

This file is where the logger stores all the error messages if any and statistics of the data the script has worked on. Below is a snippet:

error_log.log

```
2025-01-19 16:13:49,021 - INFO - Low Quality Variants (Quality <= 20): 2175
2025-01-19 16:13:49,021 - INFO - Variants that have been categorized: 8370 Variants
2025-01-19 16:13:49,021 - INFO -  - Non-coding variants: 2380
2025-01-19 16:13:49,021 - INFO -  - Synonymous variants: 1628
2025-01-19 16:13:49,021 - INFO -  - Non-Synonymous variants: 4362
2025-01-19 16:13:49,021 - INFO - Results saved: View Output_Variants.tsv and Variant_Numbers.png

```

Possible improvements:

Accommodating Edge Cases in CDS Verification:

The script works under the assumption that CDS sequences are always multiples of 3. This issue could be addressed using a designated reading frame to handle the frame shifts, ensuring there are no errors while translating the sequence.

Incorporating Functionality to Address Insertions and Deletions:

The script currently is only capable of addressing single nucleotide polymorphisms, that is, cases where there's only one reference allele and one alternate allele. In the case of insertions the alternate allele is longer than the reference allele and in the case of deletions the reference allele is longer than the alternate allele. This script works under the assumption that all mutations are single base substitutions, thus being unable to handle cases where multiple nucleotides are removed or added.

Adding this functionality will greatly increase the possible categories that the variants fall under. This can be done by adding measures to watch frameshifts while translating the nucleotide sequences, and by doing so we can better understand the impact on protein function, since proteins are structurally specific.

Biopython Warning- Partial Codons:

This warning appears when the script is run, while translating the nucleotide sequence into amino acids. Below is a snippet of the warning:

```
lochan.karthick@Lochans-MacBook-Air BCPy assignment % python3 3046323_assignment.py --vcf assessmentData.vcf --gff Plasm
oDB-54_Pfalciparum3D7.gff --fasta PlasmoDB-54_Pfalciparum3D7_Genome.fasta

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/Bio/Seq.py:2879: BiopythonWarning: Parti
al codon, len(sequence) not a multiple of three. Explicitly trim the sequence or add trailing N before translation. This
 may become an error in future.
  warnings.warn(
lochan.karthick@Lochans-MacBook-Air BCPy assignment %
```

This warning arises due to some of the CDS sequences not having a length that is a multiple of three. Since codons are read in triplets, this leads to absent amino acids in the translated sequence.

This warning could be handled by:

1. Trimming the sequence before translation, if the sequence length is not a multiple of three. But this may remove real nucleotides, thus issues in translation may still persist.
2. We could also add a symbol denoting unknown nucleotides, but we would introduce the odds of the last codon being wrong into the script.

Conclusion:

This script categorizes the variants efficiently, and provides a detailed well structuredTSV table that reports the impact of the variants on the protein sequence. The errors are logged in an organised manner in a separate file. The script at its current level is useful for understanding and studying genetic variations but the addition of capabilities to handle Indels would improve both the flexibility, accuracy and finally its ability to handle more rigorous analytical work.