**CSE 601: Data Mining and Bioinformatics**

**Project 1: Dimensionality Reduction & Association Analysis**

**Part 2: Association Analysis**

**Team Members:**

1. **Name:** Bhagutharivalan Natarajan Muthukkannu

**UB Person No:**

**UBIT Name:**

1. **Name:** Harish Kannan Venkataramanan

**UB Person No:**

**UBIT Name:** hvenkata

1. **Name:** Praveen Mohan

**UB Person No:** 50321225

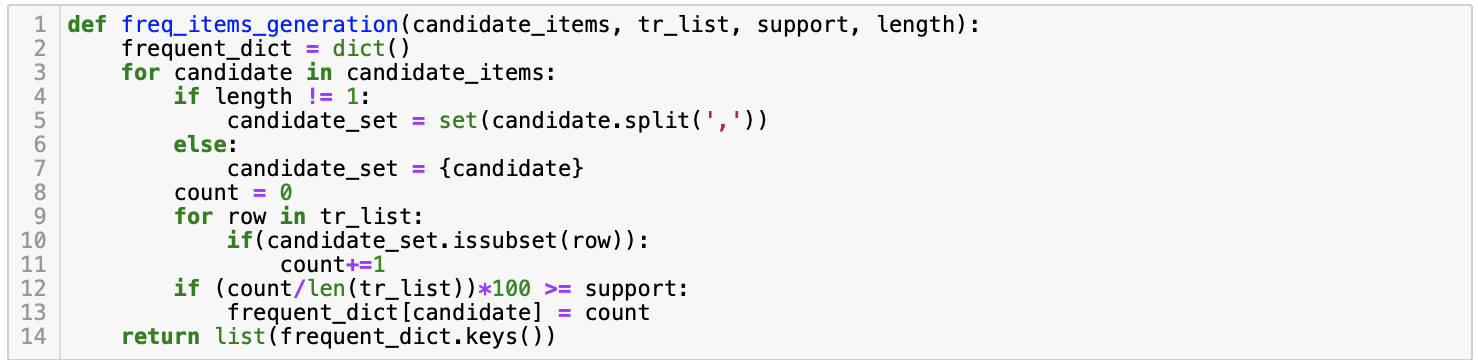
**UBIT Name:** pmohan2

**Generating Frequent Itemset using Apriori Algorithm**

**Packages used:**

1. *“import itertools”*- To iterate through each frequent itemsets to generate candoate rules.
2. *“import pandas as pd”* - To store and retrieve RULES | HEAD | BODY for obtaining template results.

**Functions implemented:**

* ***freq\_items\_generation(candidate\_items, tr\_list, support, length)***

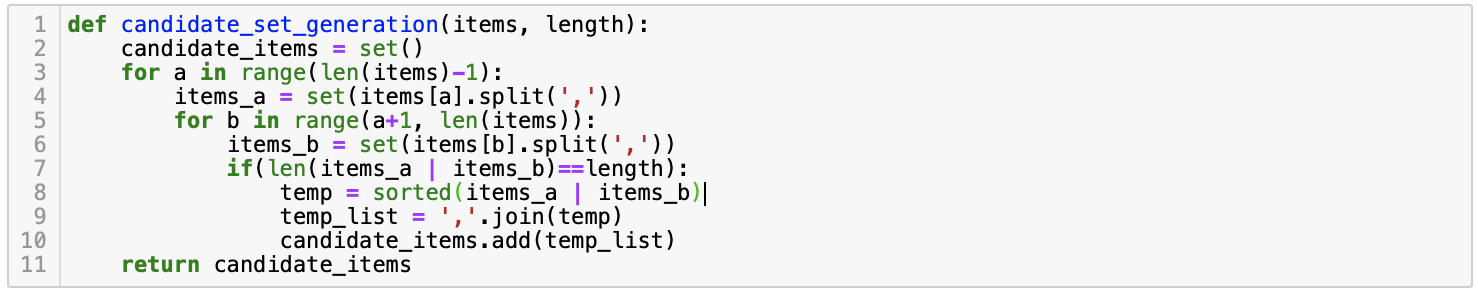
The function freq\_items\_generation(candidate\_items, tr\_list, support, length) generates a list of frequent itemset.

**Inputs:**

1. candidate\_items - A set of candidate frequent itemset for generating frequent itemsets for the corresponding threshold.
2. tr\_list - Transaction list or list of all rows from the original data to find the support count for candidate frequent itemset.
3. support - Minimum Support Threshold in percentage.
4. length - Length of the candidate frequent itemset

**Output:**

1. list(frequent\_dict.keys()) - A list of frequent itemset.

* ***candidate\_set\_generation(items, length)***

The function candidate\_set\_generation(items, length) generates a set of candidate frequent itemset.

#### Inputs:

1. items - A set of frequent itemset for generating next length candidate frequent itemset.
2. length - Length of the itemset.

#### Output:

1. candidate\_items - A set of candidate frequent itemset.

* ***open\_file(filename)***

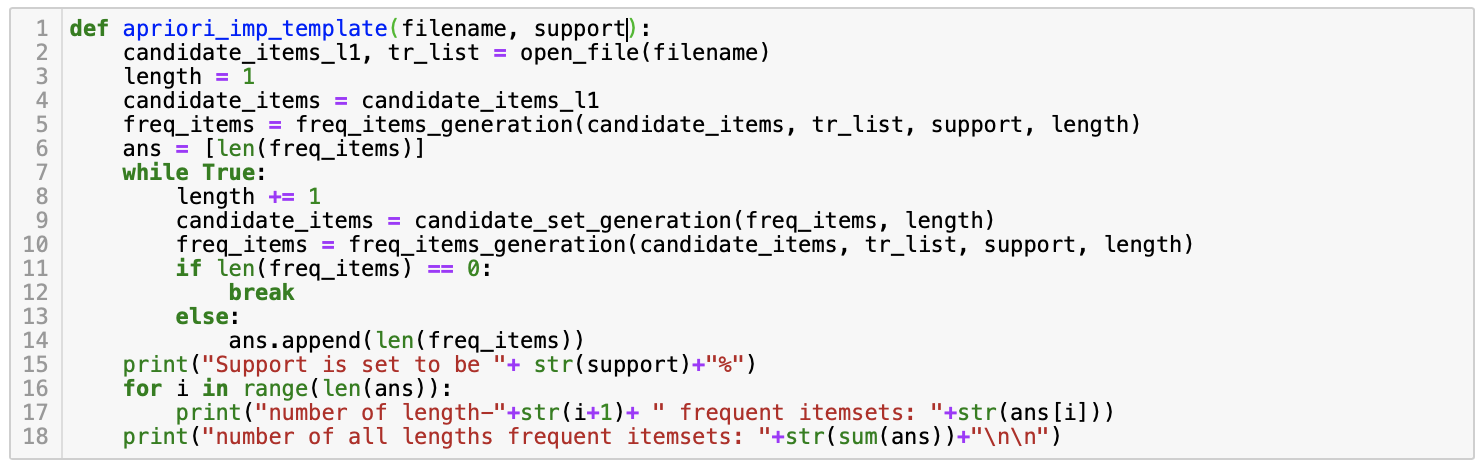
The function open\_file(filename) generates a list of candidate frequent itemset of length 1 and a transaction list or list of all rows from the original data to find the support count for candidate frequent itemset.

#### Inputs:

1. filename - Name of the original gene data file.

#### Output:

1. candidate\_items\_l1 - A list of candidate frequent itemset of length 1.
2. tr\_list - Transaction list or list of all rows from the original data to find the support count for candidate frequent itemset.

* ***def apriori\_imp\_template(filename, support)***

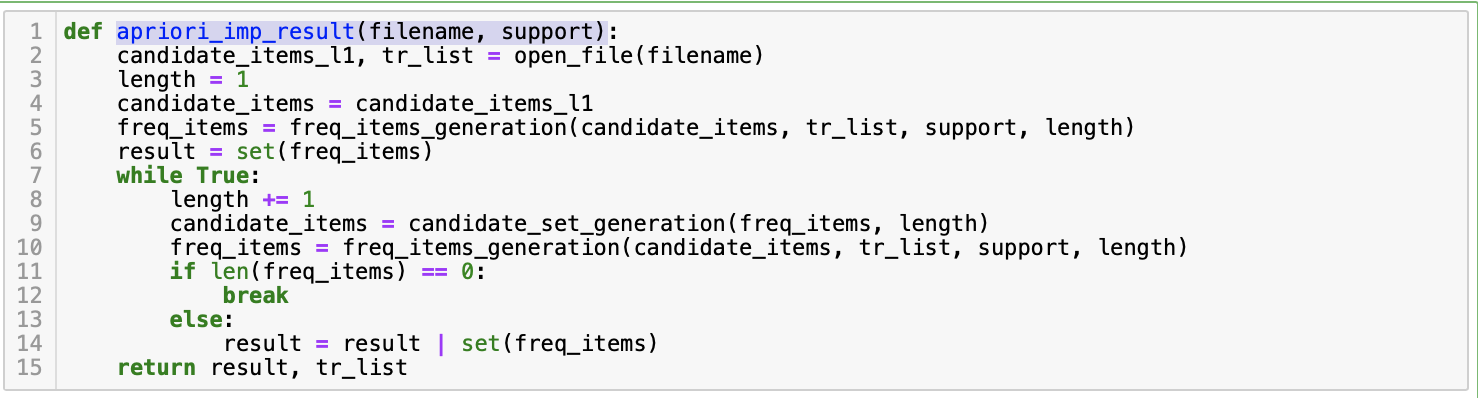
The function apriori\_imp\_template(filename, support) generates the template for part 1 of Apriori Algorithm in generating the frequent itemset for the given support.

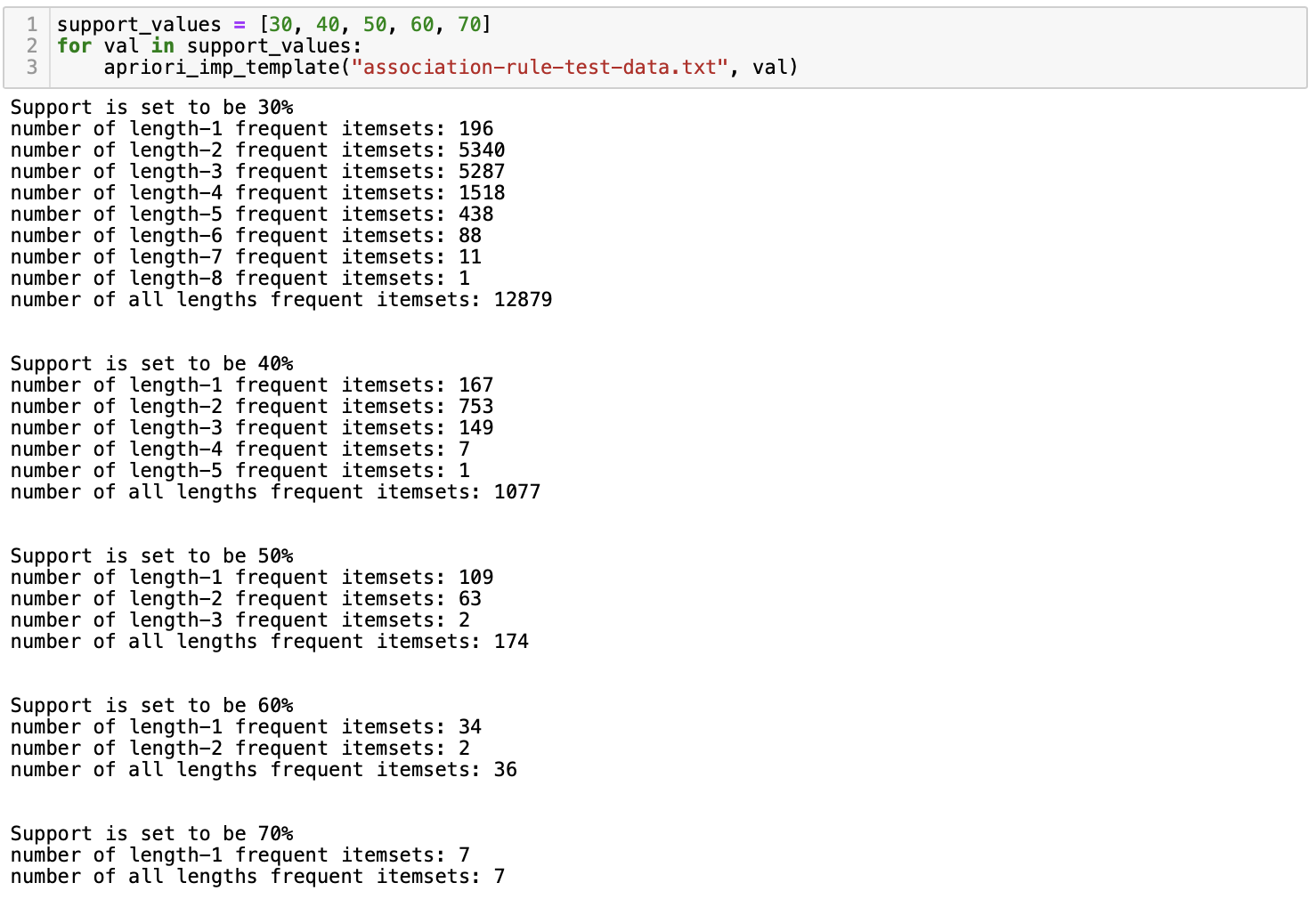
#### Inputs:

1. filename - Name of the original gene data file.
2. support - Minimum Support Threshold in percentage.

#### Output:

1. The template for part 1 of Apriori Algorithm in generating the frequent itemsets for the given support.

* ***apriori\_imp\_result(filename, support)***

**Results for given support values [30%, 40%, 50%, 60%, 70%]**

* ***freq\_count(freq\_itemset, tr\_list)***

******

The function freq\_count(freq\_itemset, tr\_list) generates the count for the given itemset in the transaction database.

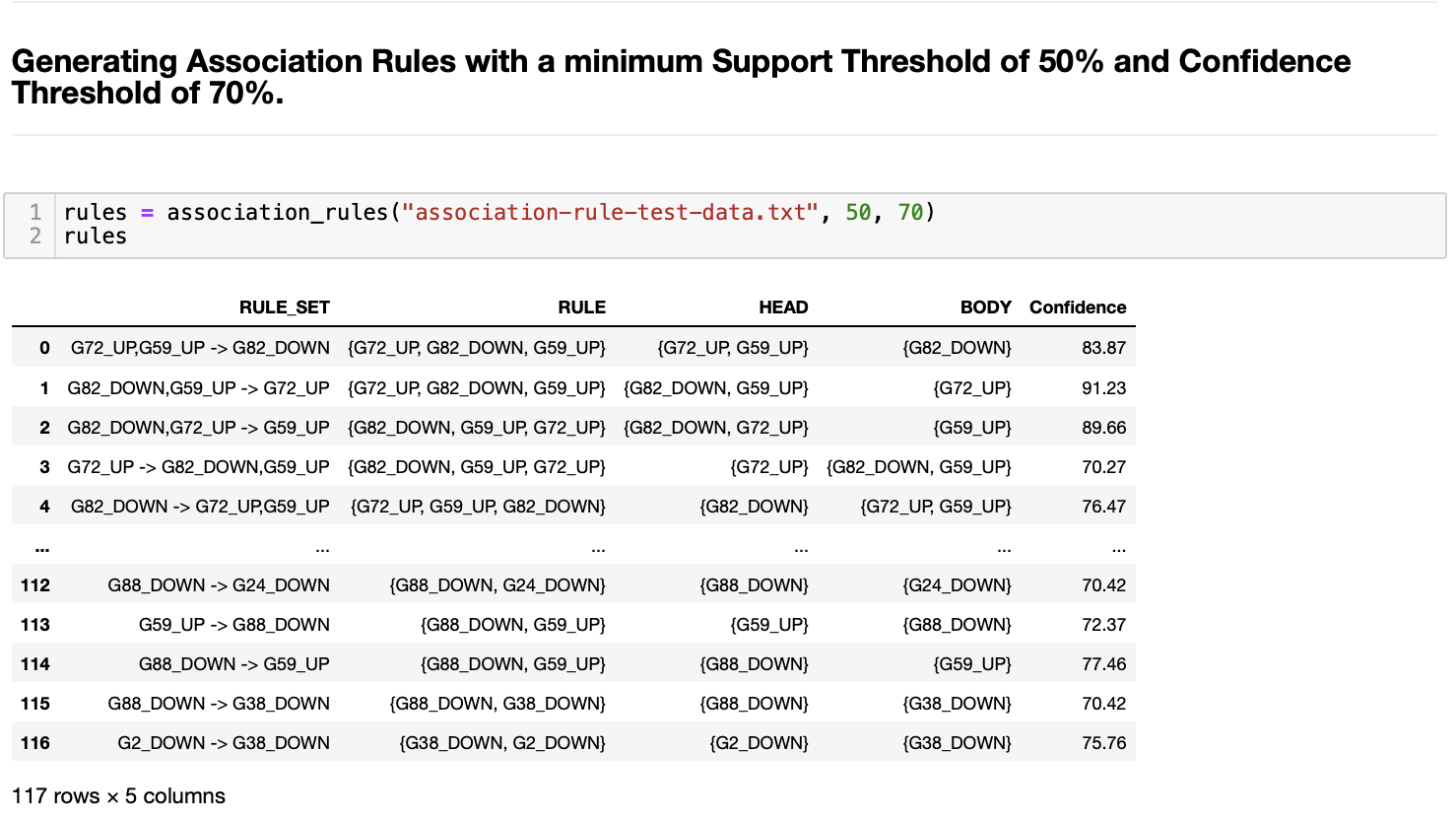
#### Inputs:

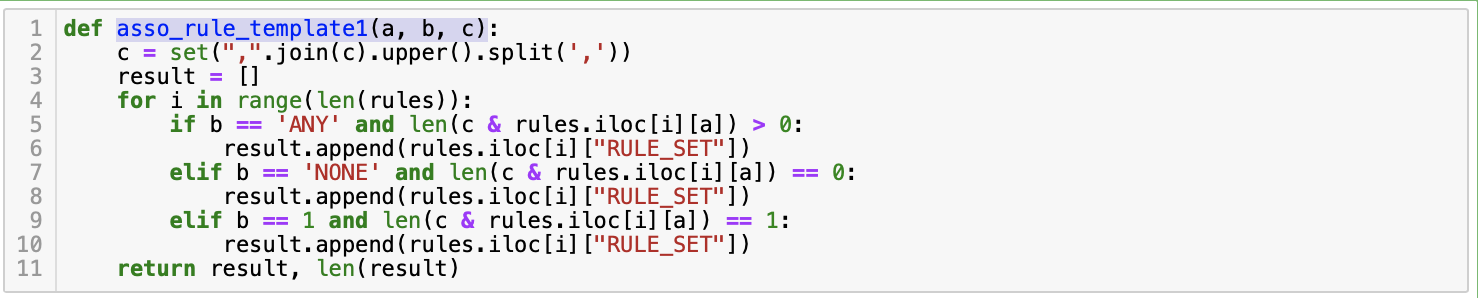
1. freq\_itemset - A frequent itemset for the given support threshold.
2. tr\_list - Transaction list or list of all rows from the original data to find the support count for candidate frequent itemset.

#### Output:

1. count = The count for the given frequent itemset in the tr\_list.

* ***association\_rules(filename, support, confidence)***

******

* ***asso\_rule\_template1(a, b, c)***

The function asso\_rule\_template1(a, b, c) generates the results for template 1 for the given query.

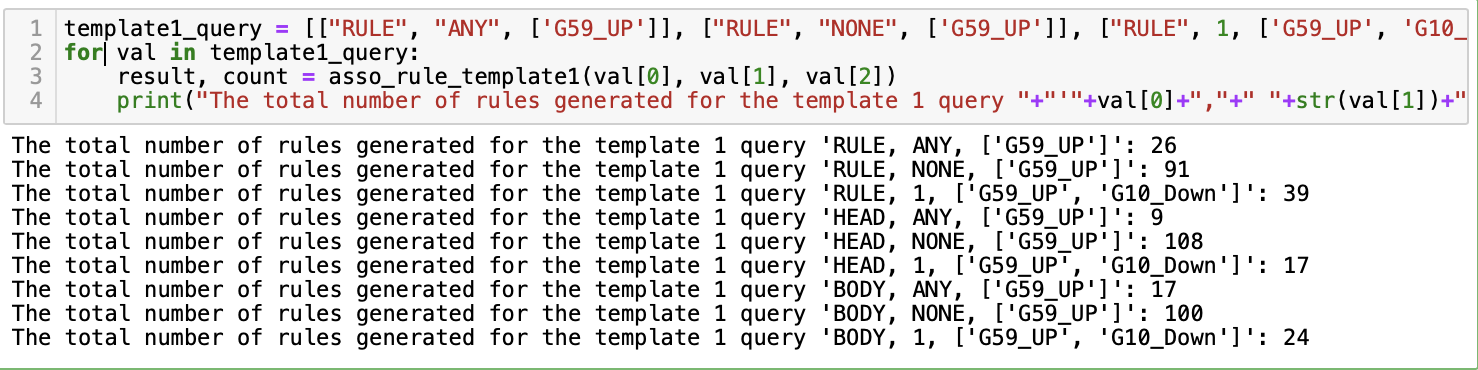
#### Inputs:

1. a - "RULE" | "HEAD" | "BODY"
2. b - "ANY" | "NONE" | 1
3. c - ["Gene", ...]

#### Output:

1. result - A list of rules for the given query.
2. len(result) - Total number of rules generated for the given query.

**Template 1 results   
(support = 50%, confidence = 70%)**



* ***asso\_rule\_template2(a, b)***

******

The function asso\_rule\_template2(a, b) generates the results for template 2 for the given query.

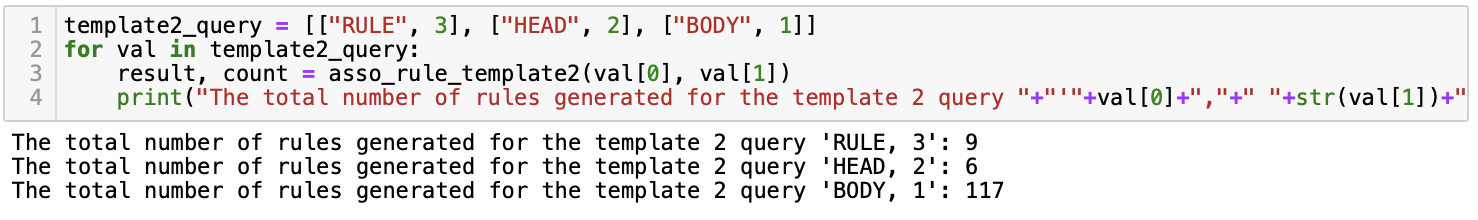
#### Inputs:

1. a - "RULE" | "HEAD" | "BODY"
2. b - integer (length)

#### Output:

1. result - A list of rules for the given query.
2. count - Total number of rules generated for the given query.

**Template 2 results**

****

* ***temp\_operator(string)***

******

The function temp\_operator(string) splits the first input for template 3 into respective template value and the corresponding operator.

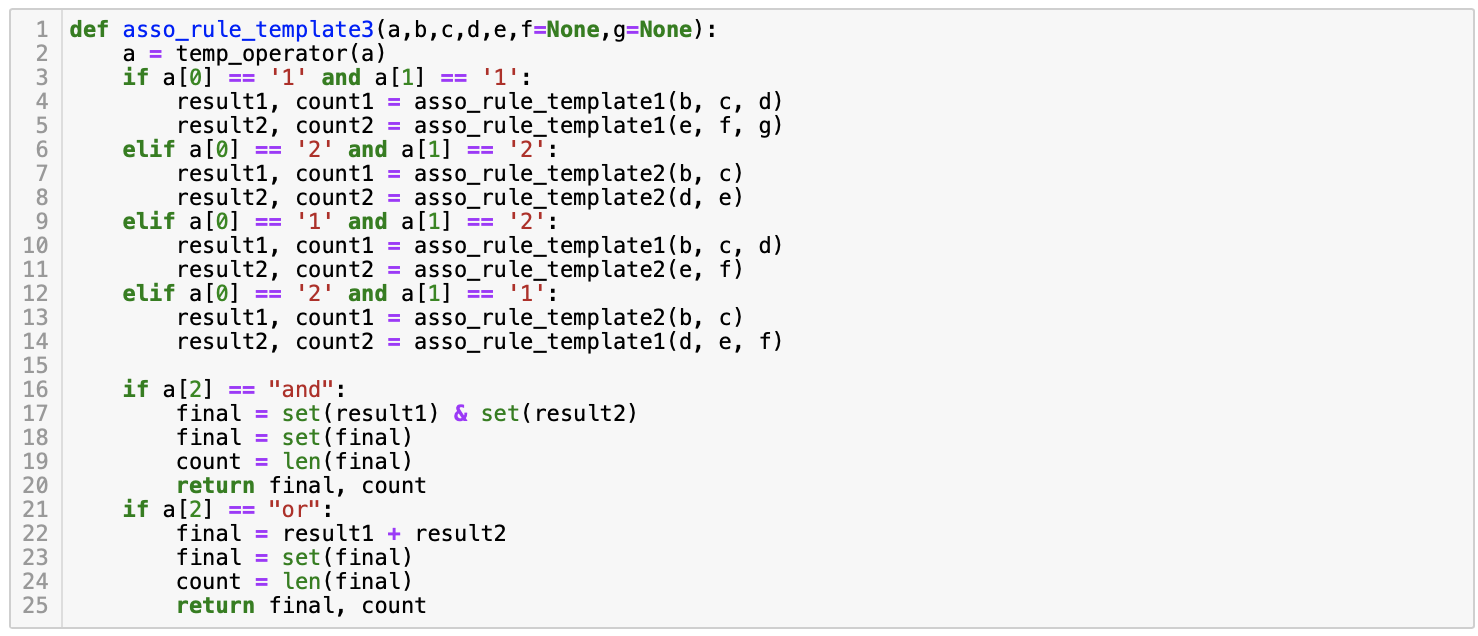
#### Inputs:

1. string - "1or1", "2or2", "1or2", "1and1", "2and2", "1and2".

#### Output:

1. list - [template number, template number, operator]

* ***asso\_rule\_template3(a,b,c,d,e,f=None,g=None)***



The function asso\_rule\_template2(a, b) generates the results for template 3 for the given query.

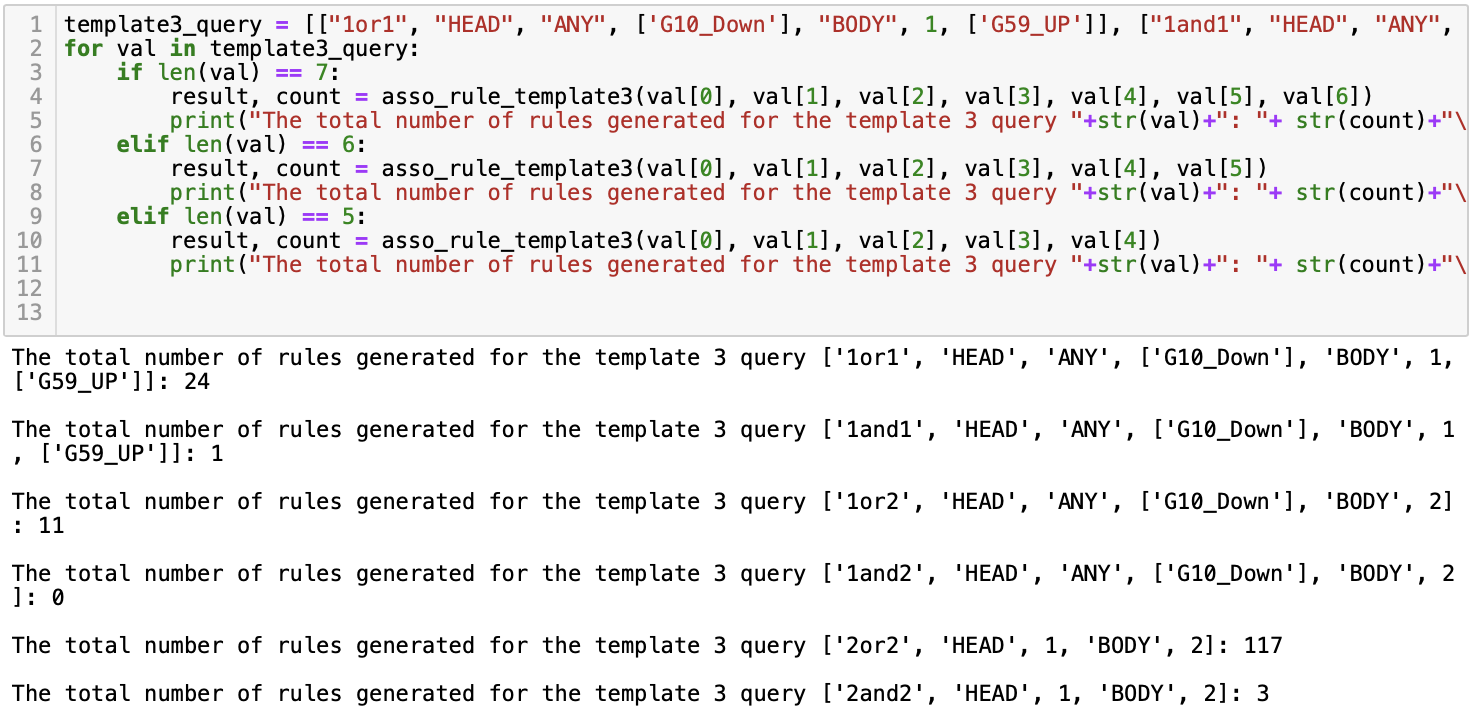
#### Inputs:

1. a - "1or1", "2or2", "1or2", "1and1", "2and2", "1and2".
2. b - "RULE" | "HEAD" | "BODY"
3. c - "ANY" | "NONE" | 1 (or) integer
4. d - ["Gene", ...] (or) "RULE" | "HEAD" | "BODY"
5. e - "RULE" | "HEAD" | "BODY" (or) "ANY" | "NONE" | 1
6. f - "ANY" | "NONE" | 1 (or) ["Gene", ...] (or) None
7. g - ["Gene", ...] (or) None

#### Output:

1. final - A set of rules for the given query.
2. count - Total number of rules generated for the given query.

**Template 3 results**

****