In [10]:
```
import pandas as pd
```

In [11]:
```
#Function that taken in input of tax_ids as a chain, and appends the parent_ta
x_id of the last element to that list
def parent_append(chain):
    if chain[-1] == 1:
        return chain
    else:
        temp = nodes.loc[nodes['tax_id'] == chain[-1]]
        chain.append(temp.iloc[0,1])
        return chain
```

In [12]:
```
#Nodes and names in same directory as python file
#https://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz
#Specifying the format nodes.dmp is in
nodes = pd.read_table('nodes.dmp', sep="\\t\|\\t", header=None,engine='python'
)
```

In [14]:
```
#Cleaning the nodes table
ignorecols = [2,3,4,5,6,7,8,9,10,11,12]
nodes.drop(nodes.columns[ignorecols],axis=1,inplace=True)
nodes.columns = ['tax_id', 'parent_tax_id'] #placing header and naming columns
to tax_id and parent_id
```

In [16]:
```
#Specifying the format names.dmp is in
names = pd.read_table('names.dmp', sep="\\t\|\\t", header=None,engine='python'
)
```

In [17]:
```
#Cleaning the names table
ignorecols = [2,3]
names.drop(names.columns[ignorecols],axis=1,inplace=True)
names.columns = ['tax_id', 'name_txt'] #placing header and naming columns to t
ax_id and name_txt
```

In [19]:
```
print("This program shows you the least common ancestor of two organisms")
#Input of organism 1
while True:
    org1 = str(input("Enter name of the first organism: ")) #Eg. H1N1 swine in
fluenza virus
    org1row = names[names['name_txt'].str.contains(org1, na = False, regex=Fal
se)] #searches input on the names_txt column
    if org1row.empty:
        print('Organism name not found in NIH database: Please try again')
    else:
        break
    pass
```

```
This program shows you the least common ancestor of two organisms
Enter name of the first organism: blrhh
Organism name not found in NIH database: Please try again
Enter name of the first organism: H1N1 swine influenza virus
```

In [22]:
```
#Input of organism 2
while True:
    org2 = str(input("Enter name of the first organism: ")) #Eg. Middle East r
espiratory syndrome-related coronavirus
    org2row = names[names['name_txt'].str.contains(org2, na = False, regex=Fal
se)] #searches input on the names_txt column
    if org2row.empty:
        print('Organism name not found in NIH database: Please try again')
    else:
        break
    pass
```

```
Enter name of the first organism: Middle East respiratory syndrome-related co
ronavirus
```

In [23]:
```
#Defaulting to the first result if there are multiple. Can be refined
org1_tax_id = org1row.iloc[0,0]
org2_tax_id = org2row.iloc[0,0]
```

In [28]:
```
q = True
while q:
    org1_chain, org2_chain = [org1_tax_id], [org2_tax_id] #initalizing a chain
with the tax_id of each organism
    if org1_chain[-1] == org2_chain[-1]: #if both organisms have same tax_id
        print('The Least Common Ancestor is: ', names.iloc[org1_chain[-1],1])
    else:
        i = 0 #Count of how many levels of the taxonomy tree traveresed
        while True:
            org1_chain = parent_append(org1_chain) #calls function to add pare
nt_tax_id
            org2_chain = parent_append(org2_chain)
            common_ancestor = list(set(org1_chain).intersection(org2_chain)) #
common ancestor is a tax_id present in both lists
            if common_ancestor != []:
                print("Found an ancestor with tax_id:", common_ancestor[-1])
                ancestor_row = names.loc[names['tax_id'] == common_ancestor[0
]] #take common ancestor row from the .dmp file
                if ancestor_row.iloc[0,1] == 'all':
                    print ('It is disjoint. The Least common ancestor is: all/
root')
                else:
                    print('The Least common ancestor is: ', ancestor_row.iloc[
0,1]) #defaults to first result. Can include alternate names if needed
                print(i+1, " levels of the tree traversed")
                break
            else:
                i += 1
                continue

    q = False
```

```
Found an ancestor with tax_id: 2732396
The Least common ancestor is:  Orthornavirae
9  levels of the tree traversed
```

In [29]:
```
#Create a list the shows the path down the tree by reversing the chains
org1_tree, org2_tree = org1_chain[::-1], org2_chain[::-1]
```

In [30]:
```
#Prints the links to the common ancestor with nicely formatted columns
print("%30s %s" % (org1_tree[0],names.loc[names.loc[names['tax_id'] == org1_tr
ee[0]].index[0],'name_txt']))
for i in range(1,min(len(org1_tree),len(org2_tree))):
    print("%-40s %s" % ((str(org1_tree[i]) + " " + names.loc[names.loc[names[
'tax_id'] == org1_tree[i]].index[0],'name_txt']),
                        (str(org2_tree[i]) + " " + names.loc[names.loc[names[
'tax_id'] == org2_tree[i]].index[0],'name_txt'])))
if len(org1_tree) > len(org2_tree):
    for i in range(len(org2_tree),len(org1_tree)):
        print((str(org1_tree[i]) + " " +
                            names.loc[names.loc[names['tax_id'] == org1_tree[i
]].index[0],'name_txt']))
if len(org2_tree) > len(org1_tree):
    for i in range(len(org1_tree),len(org2_tree)):
        print("%-40s %s" % (" ", (str(org2_tree[i]) + " " +
                                  names.loc[names.loc[names['tax_id'] == org2_
tree[i]].index[0],'name_txt'])))
```

```
                       2732396 Orthornavirae
2497569 Negarnaviricota                  2732408 Pisuviricota
2497571 Polyploviricotina                2732506 Pisoniviricetes
2497577 Insthoviricetes                  76804 Nidovirales
2499411 Articulavirales                  2499399 Cornidovirineae
11308 Orthomyxoviridae                   11118 Coronaviridae
197911 Alphainfluenzavirus               2501931 Orthocoronavirinae
11320 FLUAV                              694002 Betacoronavirus
114727 H1N1                              2509494 Merbecovirus
36420 H1N1 swine influenza virus         1335626 MERS
```