1) Instructions to run the code:

To run the code, open your terminal and go in the directory that the code is located.

1. Run the code using python 3 scrape_and_cluster.py

2. The user will be prompted to enter an integer that will define how many posts to
fetch at a time.

3. Next, the user will be asked after how much time interval the fetching must
happen again.

4. Then, the background function with data fetching and storing will start running.

5. The clustering of the text will start as soon as the data is fetched.

6. User will be asked to enter a string to be clustered, or enter 'quit' if they want the background function to stop and the code to terminate.

7. When the user enters a string, it will be assigned to a cluster.The output of the cluster number and the 5 important representative words of that cluster will be given as an output.

8. Finally, a scatterplot that visualizes all the clusters in a 3-dimensional space will be given to the user as an output.

2) Converting text to vectors:

As instructed in the questionnaire, our first task was to take relevant text from the posts and convert them into vectors of fixed dimensions that capture the meaning of the text. For this, we first sent the post title and comments that we scraped in Part 1 of this assignment through a preprocessing pipeline (Removing stop words, etc). Then, we used a doc2vec model to convert them into vectors of a foxed dimension that can be fed into a clustering model.

```
def vectorizer_trainer():
    if connection.is_connected():
        cursor = connection.cursor()
        select_query = "SELECT Post_title, Comments FROM ReBot;"
        cursor.execute(select_query)
        result = cursor.fetchall()
        concatenated_list = [(''.join(tpl)) for tpl in result]
        tagged_data = [TaggedDocument(words=word_tokenize(remove_stop_words(doc.lower())), tags=[str(i)]) for i, doc in enumerate(concatenated_list)]
        model = Doc2Vec(vector_size=20,min_count=2, epochs=50)
        model.build_vocab(tagged_data)
        model.train(tagged_data,total_examples=model.corpus_count,epochs=model.epochs)

    return concatenated_list, model
```

The output of this function is a list of lists (vectors) that depict each of the reddit posts.

3) Clustering the vectors:

For this step, we used Sklearn python library to fit a Kmeans model on our training vectors. This model takes the vectors as input and outputs one label for each vector. Since we had around 10 categories of posts in Lab 1, we decided to take number of clusters = 10.

```
def cluster_trainer(model, concatenated_list):
    a =  model.docvecs.get_normed_vectors()

    NUMBER_OF_CLUSTERS = 10
    kmeans_model = KMeans(n_clusters=NUMBER_OF_CLUSTERS)
    centroids = kmeans_model.fit(a).cluster_centers_
    doc_ids = kmeans_model.labels_
    # computes cluster Id for document vectors
    pca = PCA(n_components=3)
    a_pca = pca.fit_transform(a)

    mydata = {
        'sentence' : concatenated_list,
        'cluster_no' : doc_ids,
        'Centroid_x' : centroids[doc_ids, 0],
        'Centroid_y' : centroids[doc_ids, 1],
        'Centroid_z' : centroids[doc_ids, 2],
        'X': 0,
        'Y': 0,
        'Z': 0
    }
    df = pd.DataFrame(mydata)
    for i in range(0,len(a_pca)):
        df['X'][i] = a_pca[i][0]
        df['Y'][i] = a_pca[i][1]
        df['Z'][i] = a_pca[i][2]
    print(df,'\n\n')
    return df, kmeans_model
```

Finally, we take the original post sentences, the corresponding output label for each post sentence, and the centroids for each cluster. The output of this function is a data frame with the sentence, the cluster number assigned to it, cluster centroid co-ordinated and vector co-ordinates.

To extract the semantic meaning of each cluster at the end of the clustering, we segregate all the text in each of the clusters and put them through a keyword extraction pipeline that finds the 5 most important keywords that represent that cluster.

```
def extract_cluster_keywords(texty):
    nlp = spacy.load("en_core_web_sm")
    # Define the sentence you want to extract keywords from

    # Process the sentence with spaCy
    doc = nlp(texty)
    # Extract important keywords (nouns and adjectives) from the sentence
    keywords = [token.text for token in doc if token.pos_ in ["NOUN", "ADJ"]]
    # Count the frequency of each keyword
    keyword_counts = Counter(keywords)
    # Extract keywords that occur more than once and are therefore important
    top_10_keywords = [keyword for keyword, count in keyword_counts.most_common(5)]
    # Print the important keywords
    return top_10_keywords
```

This gives us a better intuition of what is happening in the code and what each of the clusters means.

4) Periodic fetching and updating clusters:

This task is essential to the assignment because it makes sure the database is kept updated on Reddit in real-time. For this, we have to use multi-threading. Specifically, a daemon thread is started in the background, that calls an infinitely running function.

This function fetches the latest data from Reddit, preprocesses it, stores it on our SQL database. After every iteration of this function, it sleeps for a set amount of user-defined time (approximately 5-10 minutes) and then does the same thing again.

```
daemon = Thread(target=update_database, args=(update_interval*60,), daemon=True, name='Background')
daemon.start()
print(f"\nFetching script will run every {update_interval} minute(s).")
```

```
def update_database(update_interval):
    while True:
        print('\n---Updating database---\nFetching new posts...')
        df = fetch_reddit_posts(subreddit, num_posts)
        print('\n---Updating database---\nPreprocessing and storing fetched data...')
        newdf = preprocess_reddit(df)
        storage(newdf)

        sleep(update_interval)
```

This daemon thread will continue running until the user inputs ‘quit’ or the code is terminated.

5) Clustering a new user input:

Finally, after all the text scraping, preprocessing, and clustering, the code takes a new user input. This input is a sentence, that is now vectorized by the same doc2vec model that vectorized all the other training sentences.

```
def vectorizer(data,model):
    new_vector = model.infer_vector(word_tokenize(remove_stop_words(data.lower())))

    return new_vector
```

Further, it is predicted by the previously trained Means model and the assigned cluster number is given as an output.

To give the user a better understanding of the cluster that the sentence is assigned to, the 5 representative keywords of that cluster are printed along with the output cluster number.

Example 1:

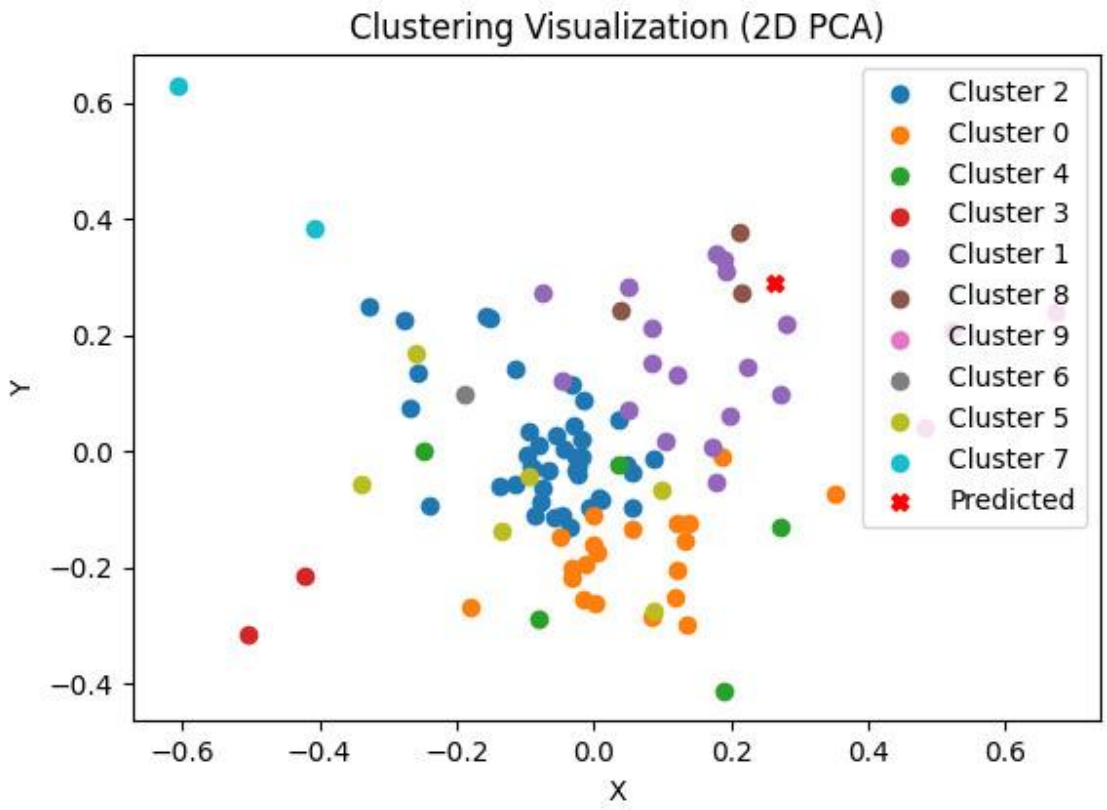
```python
def clusterizer(new_vector, kmeans_model):
    predicted_cluster = kmeans_model.predict(new_vector.reshape(1, -1))[0]
    return predicted_cluster
```

```
Enter number of posts to fetch : 1000

---Updating database---
Fetching new posts...

Fetching script will run every 360 minute(s).

Enter text to cluster or 'quit' to stop: ^[[C
---Updating database---
Preprocessing and storing fetched data...

---Updating database---
Resolving duplication...

---Updating database---
Sent to server :)
air is above water in which we can fly in the aerospace using an aeroplane?

---Training Model---
Vectorizing training data

---Training Model---
Clustering training data
```

```
---Clustering User Input---
User input vectorizing

---Clustering User Input---
User input clustering

---Results---
Prediction for  ir is above water in which we can fly in the aerospace using an aeroplane?  is  8
The keywords of this cluster are ['windows', 'heat', 'way', 'insulation', 'years']
[-0.01886214 -0.12480074 -0.06552979  0.14331028 -0.3859464   0.06920452
  0.03612867  0.01143242 -0.5456247  -0.01144715 -0.13158733 -0.0380294
  0.11832552 -0.0197201   0.29690224  0.03816007  0.18431973 -0.20183325
 -0.439343   -0.29367104]
[[0.2629825 0.2905133]]
```

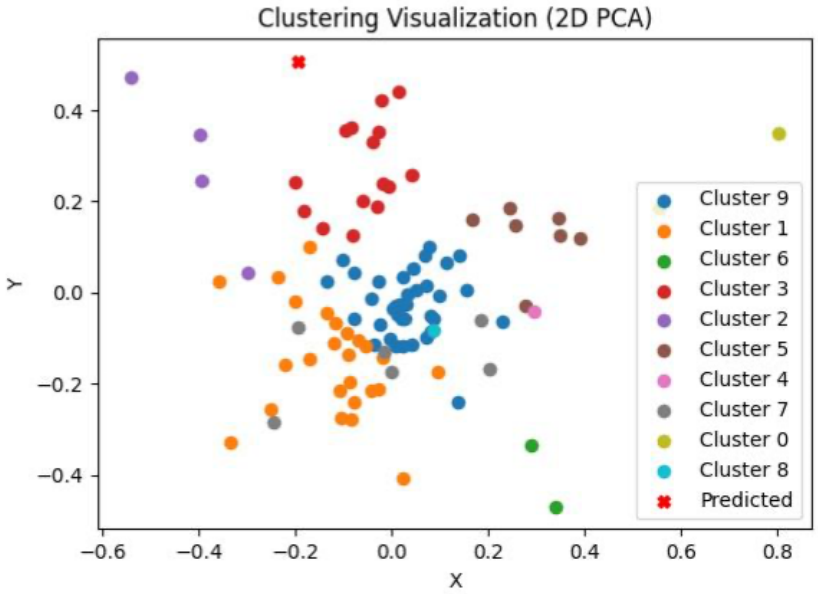
As printed above, the predicted cluster for this sentence is cluster 8 (brown points)

This is a scatterplot as a visual representation of all the clusters formed by the model. The predicted cluster for this user input is 8. We can see that the predicted cross is in the area with cluster 8 (brown).


Clustering Visualization (2D PCA)
Cluster 2
Cluster 0
Cluster 4
Cluster 3
Cluster 1
Cluster 8
Cluster 9
Cluster 6
Cluster 5
Cluster 7
Predicted
0.6
0.4
0.2
0.0
−0.2
−0.4
Y
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
X

Example 2:

```
water is filled with waste and oil spillage

Enter text to cluster or 'quit' to stop:
---Training Model---
Vectorizing training data

---Training Model---
Clustering training data
                                        sentence  cluster_no  Centroid_x  Centroid_y         X         Y
0   this robotic exoskeleton can help runners spri...           9    0.031579    0.074606  0.032681 -0.002514
1   this octopusinspired patch could deliver drugs...           1    0.102270    0.077351 -0.077599 -0.239906
2   scientists develop gel from spider venom to bo...           9    0.031579    0.074606  0.024570 -0.118459
3   we try out the first legal level  automated dr...           1    0.102270    0.077351 -0.334469 -0.328432
4   artificial wombs for premature babies might so...           6    0.140081   -0.065842  0.340253 -0.469420
..                                                ...         ...         ...         ...       ...       ...
95  a new spaceflight startup wants to use a wild ...           1    0.102270    0.077351  0.023750 -0.407944
96  computer chip with builtin human brain tissue ...           9    0.031579    0.074606 -0.023330 -0.071287
97  scientists invent doublesided solar panel that...           3   -0.032839    0.149678 -0.201309  0.241332
98  new hope for resurrecting extinct plants with ...           9    0.031579    0.074606  0.013859 -0.042951
99  scientists develop stretchable robotic fabrics...           9    0.031579    0.074606  0.068374  0.080527

[100 rows x 6 columns]



---Clustering User Input---
User input vectorizing

---Clustering User Input---
User input clustering

---Results---
Prediction for  water is filled with waste and oil spillage  is  3
The keywords of this cluster are ['power', 'light', 'energy', 'battery', 'windows']
[-0.06571178  0.04241375 -0.03801914  0.13361746 -0.19404286  0.04704551
 -0.21746844 -0.05502024 -0.48187003  0.12715057 -0.25365913 -0.16038461
  0.11774882 -0.00130153  0.27887335  0.12804686  0.26347727 -0.01167291
 -0.3799954  -0.12220594]
[[-0.19511054  0.50831896]]

Enter text to cluster or 'quit' to stop: quit
```



The predicted cluster for this user input is 3. We can see that the predicted cross is in the area with cluster 3 (red).