```
import numpy as np

# Create dummy word embeddings (e.g., a dictionary mapping words to vectors)
# In a real scenario, you would load these from a pre-trained model file.
word_embeddings = {
    'hello': np.random.rand(10),
    'world': np.random.rand(10),
    'python': np.random.rand(10),
    'data': np.random.rand(10),
    'science': np.random.rand(10)
}

print(f"Loaded {len(word_embeddings)} word embeddings.")
```



```
Loaded 5 word embeddings.
```

```
# Print vocabulary size
vocabulary_size = len(word_embeddings)
print(f"Vocabulary Size: {vocabulary_size}")
```

```
Vocabulary Size: 5
```

```
# Display one example vector
example_word = 'python'
if example_word in word_embeddings:
    print(f"Vector for '{example_word}':\n{word_embeddings[example_word]}")
else:
    print(f"'{example_word}' not found in vocabulary.")
```

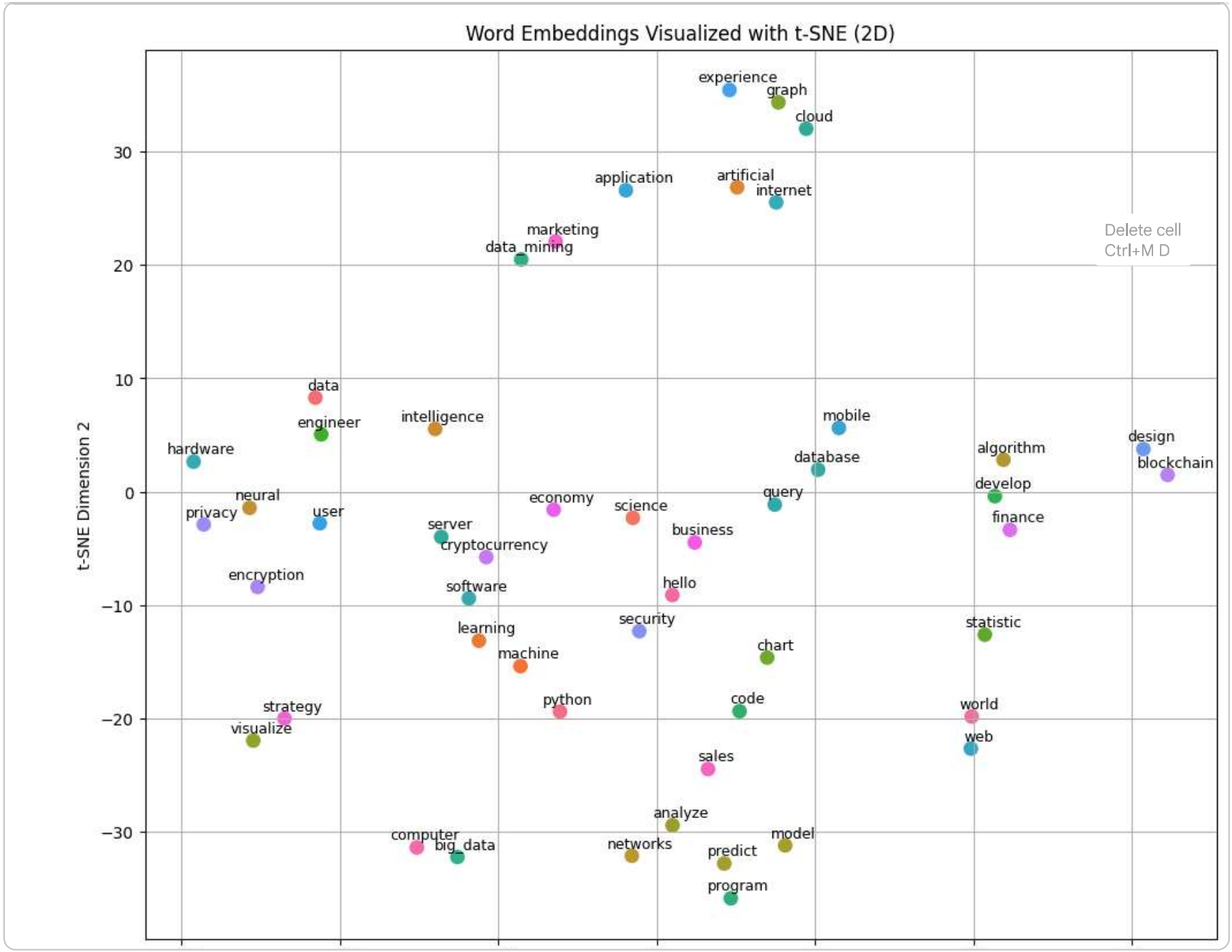
```
Vector for 'python':
[0.2696603  0.92109218 0.56760283 0.15646764 0.03632257 0.88508709
 0.66266536 0.53852002 0.78268214 0.06686785]
```

```
plt.figure(figsize=(12, 10))
sns.scatterplot(x=two_dim_vectors[:, 0], y=two_dim_vectors[:, 1], hue=actual_selected_words, legend=False, s=1

for i, word in enumerate(actual_selected_words):
    plt.annotate(word, (two_dim_vectors[i, 0], two_dim_vectors[i, 1]), textcoords="offset points", xytext=(5,5

plt.title('Word Embeddings Visualized with t-SNE (2D)')
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.grid(True)
plt.show()
```

Delete cell
Ctrl+M D



Delete cell
Ctrl+M D

```
# Initialize t-SNE model with 2 components
tsne = TSNE(n_components=2, random_state=42, perplexity=min(5, len(vectors_array) - 1))

# Fit and transform the vectors to 2 dimensions
two_dim_vectors = tsne.fit_transform(vectors_array)

print(f"Original vectors shape: {vectors_array.shape}")
print(f"Reduced vectors shape (t-SNE): {two_dim_vectors.shape}")
```

```
Original vectors shape: (50, 10)
Reduced vectors shape (t-SNE): (50, 2)
```

Delete cell
Ctrl+M D

```
# Expand dummy word embeddings
# In a real scenario, these would come from your pre-trained model
new_words = {
    'computer': np.random.rand(10),
    'science': np.random.rand(10),
    'machine': np.random.rand(10),
    'learning': np.random.rand(10),
    'artificial': np.random.rand(10),
    'intelligence': np.random.rand(10),
    'neural': np.random.rand(10),
    'networks': np.random.rand(10),
    'algorithm': np.random.rand(10),
    'model': np.random.rand(10),
    'predict': np.random.rand(10),
    'analyze': np.random.rand(10),
    'visualize': np.random.rand(10),
    'graph': np.random.rand(10),
    'chart': np.random.rand(10),
    'statistic': np.random.rand(10),
    'engineer': np.random.rand(10),
    'develop': np.random.rand(10),
    'code': np.random.rand(10),
    'program': np.random.rand(10),
    'data_mining': np.random.rand(10),
    'big_data': np.random.rand(10),
    'cloud': np.random.rand(10),
```

```
    'server': np.random.rand(10),
    'database': np.random.rand(10),
    'query': np.random.rand(10),
    'software': np.random.rand(10),
    'hardware': np.random.rand(10),
    'internet': np.random.rand(10),
    'web': np.random.rand(10),
    'mobile': np.random.rand(10),
    'application': np.random.rand(10),
    'user': np.random.rand(10),
    'experience': np.random.rand(10),
    'design': np.random.rand(10),
    'security': np.random.rand(10),
    'privacy': np.random.rand(10),
    'encryption': np.random.rand(10),
    'decryption': np.random.rand(10),
    'blockchain': np.random.rand(10),
    'cryptocurrency': np.random.rand(10),
    'finance': np.random.rand(10),
    'economy': np.random.rand(10),
    'business': np.random.rand(10),
    'strategy': np.random.rand(10),
    'marketing': np.random.rand(10),
    'sales': np.random.rand(10)
}

word_embeddings.update(new_words)

print(f"Updated total word embeddings to: {len(word_embeddings)}")
```



```
Updated total word embeddings to: 51
```

```
# Choose 30-50 meaningful words from the updated vocabulary
selected_words = [
    'python', 'data', 'science', 'machine', 'learning', 'artificial',
    'intelligence', 'neural', 'networks', 'algorithm', 'model', 'predict',
    'analyze', 'visualize', 'graph', 'chart', 'statistic', 'engineer',
    'develop', 'code', 'program', 'data_mining', 'big_data', 'cloud',
    'server', 'database', 'query', 'software', 'hardware', 'internet'
```

```
    'server', 'database', 'query', 'software', 'hardware', 'internet',
    'web', 'mobile', 'application', 'user', 'experience', 'design',
    'security', 'privacy', 'encryption', 'blockchain', 'cryptocurrency',
    'finance', 'economy', 'business', 'strategy', 'marketing', 'sales',
    'computer', 'hello', 'world'
]

# Extract corresponding vectors
vectors = []
actual_selected_words = []

for word in selected_words:
    if word in word_embeddings:
        vectors.append(word_embeddings[word])
        actual_selected_words.append(word)
    else:
        print(f"Warning: '{word}' not found in word_embeddings. Skipping.")

# Convert the list of vectors to a NumPy array
vectors_array = np.array(vectors)

print(f"\nSelected {len(actual_selected_words)} words and extracted {len(vectors_array)} vectors.")
print(f"Shape of extracted vectors array: {vectors_array.shape}")
```



```
Selected 50 words and extracted 50 vectors.
```