importing required package

In [3]:
```
import requests
import pandas as pd
from bs4 import BeautifulSoup
```

Request the page

In [7]:
```
url = "https://quotes.toscrape.com/"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")

#Extract titles
#title in this page use tag h1
titles = [soup.find("h1").get_text(strip=True)]
```

Extracting titles

In [7]:
```
#title in this page use tag h1
titles = [soup.find("h1").get_text(strip=True)]
```

In [11]:
```
print(titles)
```

```
['Quotes to Scrape']
```

extracting quotes and authors of each quote

In [13]:
```
quote_blocks = soup.find_all("div", class_="quote")

quotes = []
authors = []

for block in soup.find_all("div", class_="quote"):
    quote = block.find("span", class_="text").get_text(strip=True)
    author = block.find("small", class_="author").get_text(strip=True)


    quotes.append(quote)
    authors.append(author)
```

In [15]:
```
print(quotes)
print(authors)
```

```
['“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinkin
g.”', '“It is our choices, Harry, that show what we truly are, far more than our abilities.”', '“There are only two w
ays to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.”', '“Th
e person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”', ""“Imperfectio
n is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”"", '“Try not to be
come a man of success. Rather become a man of value.”', '“It is better to be hated for what you are than to be loved
for what you are not.”', ""“I have not failed. I've just found 10,000 ways that won't work.”"", ""“A woman is like a tea
bag; you never know how strong it is until it's in hot water.”"", '“A day without sunshine is like, you know, nigh
t.”']
['Albert Einstein', 'J.K. Rowling', 'Albert Einstein', 'Jane Austen', 'Marilyn Monroe', 'Albert Einstein', 'André Gid
e', 'Thomas A. Edison', 'Eleanor Roosevelt', 'Steve Martin']
```

create dataFrame

In [19]:
```
texts_col = titles + quotes
types_col = ["Title"] * len(titles) + ["Quote"] * len(quotes)
author_col = [None] * len(titles) + authors
df = pd.DataFrame({
    "Text": texts_col,
    "Type": types_col,
    "Author": author_col
})
```

In [21]:
```
df.head() #show the first lines of dataframe
```

Out[21]:

	Text	Type	Author
0	Quotes to Scrape	Title	None
1	“The world as we have created it is a process ...	Quote	Albert Einstein
2	“It is our choices, Harry, that show what we t...	Quote	J.K. Rowling
3	“There are only two ways to live your life. On...	Quote	Albert Einstein
4	“The person, be it gentleman or lady, who has ...	Quote	Jane Austen

add word count to dataframe

In [27]:
```
df["Word_Count"] = df["Text"].apply(lambda x: len(x.split()))

df.head()
```

Out[27]:

	Text	Type	Author	Word_Count
0	Quotes to Scrape	Title	None	3
1	“The world as we have created it is a process ...	Quote	Albert Einstein	21
2	“It is our choices, Harry, that show what we t...	Quote	J.K. Rowling	16
3	“There are only two ways to live your life. On...	Quote	Albert Einstein	26
4	“The person, be it gentleman or lady, who has ...	Quote	Jane Austen	19

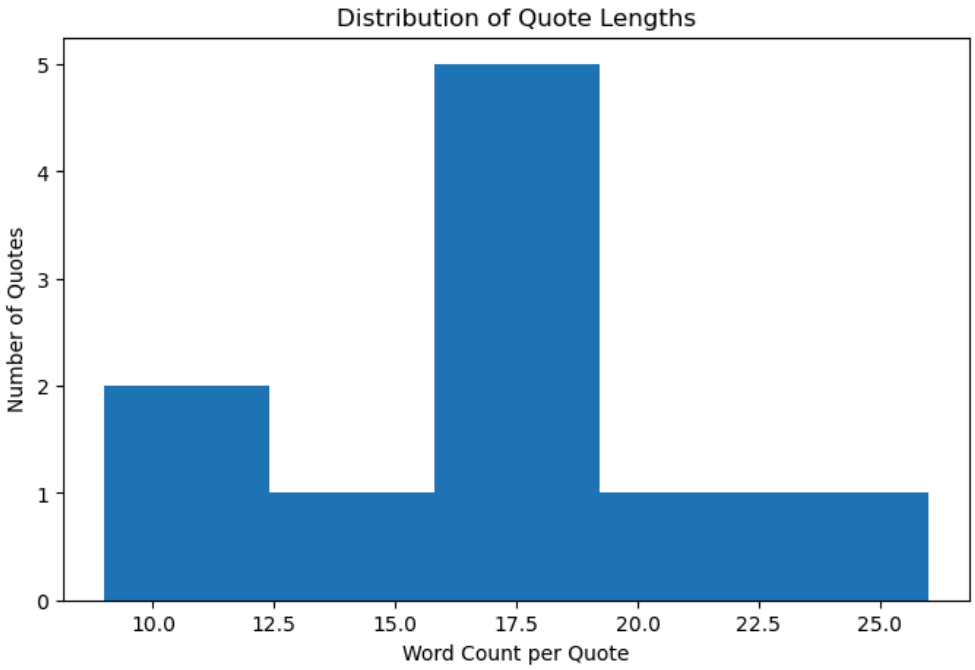
Visualization

In [33]:
```
import matplotlib.pyplot as plt

# Keep only quotes
quotes_df = df[df["Type"] == "Quote"]

plt.figure(figsize=(8,5))
plt.hist(quotes_df["Word_Count"], bins=5)

plt.xlabel("Word Count per Quote")
plt.ylabel("Number of Quotes")
plt.title("Distribution of Quote Lengths")
plt.show()
```



extract the more 5 frequents words for each author

we will use mapereduce method from the last chapter

In [68]:
```python
#the same mapper as the last homework with input text and author
import re

def mapp(text_input, author):
    """
    Maps text into ((author, word), 1) tuples
    """
    mapped = []

    # Ensure input is string and lowercase
    text = str(text_input).lower()

    # Remove punctuation
    text = re.sub(r'[^\w\s]', '', text)

    # Split into words
    words = text.split()

    # Create tuples for each word
    for word in words:
        mapped.append(((author, word), 1))

    return mapped

mapped_all = []

for _, row in quotes_df.iterrows():
    author = row["Author"]
    text = row["Text"]

    mapped_all.extend(mapp(text, author))

print(mapped_all[:20])  # preview first 10 mapped tuples
```

```
[(('Albert Einstein', 'the'), 1), (('Albert Einstein', 'world'), 1), (('Albert Einstein', 'as'), 1), (('Albert Einste
in', 'we'), 1), (('Albert Einstein', 'have'), 1), (('Albert Einstein', 'created'), 1), (('Albert Einstein', 'it'),
1), (('Albert Einstein', 'is'), 1), (('Albert Einstein', 'a'), 1), (('Albert Einstein', 'process'), 1), (('Albert Ein
stein', 'of'), 1), (('Albert Einstein', 'our'), 1), (('Albert Einstein', 'thinking'), 1), (('Albert Einstein', 'it'),
1), (('Albert Einstein', 'cannot'), 1), (('Albert Einstein', 'be'), 1), (('Albert Einstein', 'changed'), 1), (('Alber
t Einstein', 'without'), 1), (('Albert Einstein', 'changing'), 1), (('Albert Einstein', 'our'), 1)]
```

In [84]:
```python
from collections import defaultdict

#shuffle
reduced = defaultdict(list)
for key, value in mapped_all:
    reduced[key].append(value)

#reduce sum counts for each (author, word)
reduced_counts = {key: sum(values) for key, values in reduced.items()}

#words per author
author_word_freq = defaultdict(list)
for (author, word), count in reduced_counts.items():
    author_word_freq[author].append((word, count))

#Sort top 5 words per author
for author in author_word_freq:
    author_word_freq[author] = sorted(author_word_freq[author], key=lambda x: x[1], reverse=True)[:5]
```

Display results

In [88]:
```python
# 4. Display results
for author, top_words in author_word_freq.items():
    print(f"{author}: {top_words}")
```

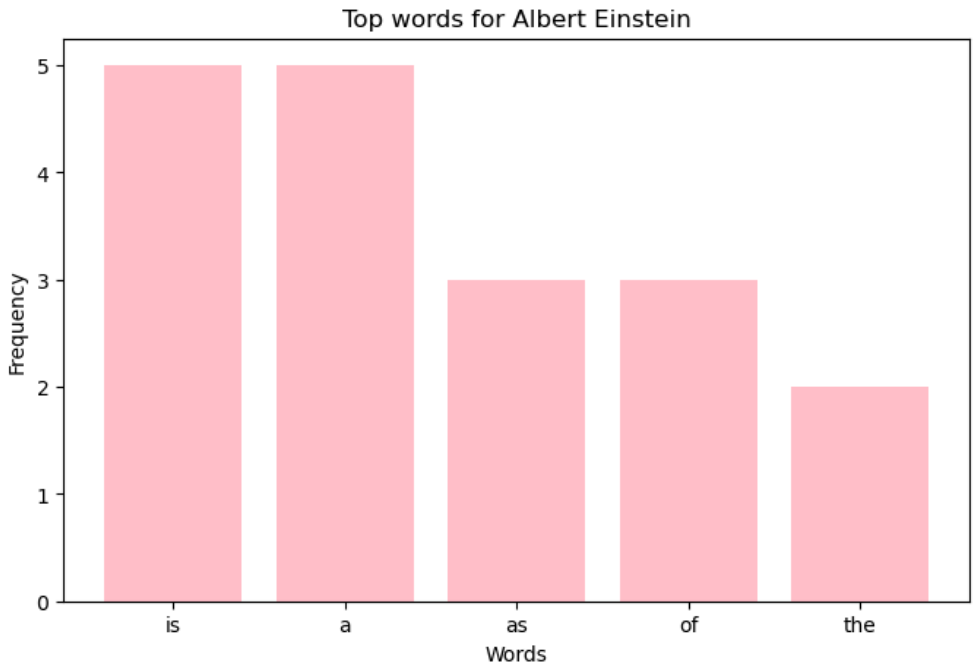
```
Albert Einstein: [('is', 5), ('a', 5), ('as', 3), ('of', 3), ('the', 2)]
J.K. Rowling: [('our', 2), ('it', 1), ('is', 1), ('choices', 1), ('harry', 1)]
Jane Austen: [('be', 2), ('the', 1), ('person', 1), ('it', 1), ('gentleman', 1)]
Marilyn Monroe: [('is', 2), ('absolutely', 2), ('imperfection', 1), ('beauty', 1), ('madness', 1)]
André Gide: [('to', 2), ('be', 2), ('for', 2), ('what', 2), ('you', 2)]
Thomas A. Edison: [('i', 1), ('have', 1), ('not', 1), ('failed', 1), ('ive', 1)]
Eleanor Roosevelt: [('a', 2), ('is', 2), ('woman', 1), ('like', 1), ('tea', 1)]
Steve Martin: [('a', 1), ('day', 1), ('without', 1), ('sunshine', 1), ('is', 1)]
```

visualization

In [96]:
```
author = 'Albert Einstein'
top_words = author_word_freq[author]

words = [w for w, c in top_words]
counts = [c for w, c in top_words]

plt.figure(figsize=(8,5))
plt.bar(words, counts, color='pink')
plt.title(f"Top words for {author}")
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.show()
```



In [ ]: