

```
# test word list
test_words = [
    "running",
    "jumped",
    "swimming",
    "swimmer",
    "apples",
    "applesauce",
    "programming",
    "programmed",
    "computers",
    "computing",
    "fascinating",
    "fascinated",
    "universality",
    "universities",
    "difficulties",
    "difficulty",
    "categorize",
    "categorization",
    "categorizing",
    "jumps",
    "laughing",
    "laughter",
    "enjoyment",
    "enjoyable",
    "enjoyment"
]
```

▼ Using Regular expressions

```
import re

def porter_stem(word):
    # Define regular expression patterns for additional rules
    ed_or_ing = re.compile(r"(.*)(ed|ing)$")
    sses_or_ies = re.compile(r"(.*)(sses|ies)$")
    at_or_bl_or_iz = re.compile(r"(.*)(at|bl|iz)$")
    y_preceded_by_consonant = re.compile(r"(.*[aeiouy])(y)$")
    y_preceded_by_vowel = re.compile(r"(.*[aeiouy])(y)$")
    replace_ational_with_ate = re.compile(r"(.*)(ational)$")
    replace_tional_with_tion = re.compile(r"(.*)(tional)$")
    replace_iveness_with_ive = re.compile(r"(.*)(iveness)$")
    replace_fulness_with_ful = re.compile(r"(.*)(fulness)$")
    replace_ousness_with_ous = re.compile(r"(.*)(ousness)$")
    replace_ality_with_al = re.compile(r"(.*)(ality)$")
    replace_icate_with_ic = re.compile(r"(.*)(icate)$")
    replace_ative_with = re.compile(r"(.*)(ative)$")
    replace_alize_with_al = re.compile(r"(.*)(alize)$")
    replace_iciti_with_ic = re.compile(r"(.*)(iciti)$")
    replace_ical_with_ic = re.compile(r"(.*)(ical)$")
    replace_ful_with = re.compile(r"(.*)(ful)$")
    replace_neous_with = re.compile(r"(.*)(eous)$")
    replace_ize_with = re.compile(r"(.*)(ize)$")
    replace_sion_with_s = re.compile(r"(.*)(sion)$")
    replace_tion_with_t = re.compile(r"(.*)(tion)$")
    replace_ence_with = re.compile(r"(.*)(ence)$")
    replace_ance_with = re.compile(r"(.*)(ance)$")
    replace_er_with = re.compile(r"(.*)(er)$")
    replace_ly_with = re.compile(r"(.*)(ly)$")
    replace_ment_with = re.compile(r"(.*)(ment)$")
    replace_able_with = re.compile(r"(.*)(able)$")
    replace_ible_with = re.compile(r"(.*)(ible)$")
    replace_ize_with = re.compile(r"(.*)(ize)$")
    replace_ion_with = re.compile(r"(.*)(ion)$")

    # Apply additional stemming rules
    if re.match(ed_or_ing, word):
        word = re.sub(ed_or_ing, r"\1", word)
    elif re.match(sses_or_ies, word):
        word = re.sub(sses_or_ies, r"\1", word)
    elif re.match(at_or_bl_or_iz, word):
        word = re.sub(at_or_bl_or_iz, r"\1", word)
    elif re.match(y_preceded_by_consonant, word):
        word = re.sub(y_preceded_by_consonant, r"\1", word)
    elif re.match(y_preceded_by_vowel, word):
        word = re.sub(y_preceded_by_vowel, r"\1y", word)
    elif re.match(replace_ational_with_ate, word):
        word = re.sub(replace_ational_with_ate, r"\1ate", word)
    elif re.match(replace_tional_with_tion, word):
        word = re.sub(replace_tional_with_tion, r"\1tion", word)
    elif re.match(replace_iveness_with_ive, word):
        word = re.sub(replace_iveness_with_ive, r"\1live", word)
    elif re.match(replace_fulness_with_ful, word):
```

```

    word = re.sub(replace_fulness_with_ful, r"\1ful", word)
elif re.match(replace_ousness_with_ous, word):
    word = re.sub(replace_ousness_with_ous, r"\1ous", word)
elif re.match(replace_ality_with_al, word):
    word = re.sub(replace_ality_with_al, r"\1al", word)
elif re.match(replace_icate_with_ic, word):
    word = re.sub(replace_icate_with_ic, r"\1ic", word)
elif re.match(replace_ative_with, word):
    word = re.sub(replace_ative_with, r"\1", word)
elif re.match(replace_alize_with_al, word):
    word = re.sub(replace_alize_with_al, r"\1al", word)
elif re.match(replace_iciti_with_ic, word):
    word = re.sub(replace_iciti_with_ic, r"\1ic", word)
elif re.match(replace_ical_with_ic, word):
    word = re.sub(replace_ical_with_ic, r"\1ic", word)
elif re.match(replace_ful_with, word):
    word = re.sub(replace_ful_with, r"\1", word)
elif re.match(replace_neous_with, word):
    word = re.sub(replace_neous_with, r"\1", word)
elif re.match(replace_ize_with, word):
    word = re.sub(replace_ize_with, r"\1", word)
elif re.match(replace_sion_with_s, word):
    word = re.sub(replace_sion_with_s, r"\1", word)
elif re.match(replace_tion_with_t, word):
    word = re.sub(replace_tion_with_t, r"\1", word)
elif re.match(replace_ence_with, word):
    word = re.sub(replace_ence_with, r"\1", word)
elif re.match(replace_ance_with, word):
    word = re.sub(replace_ance_with, r"\1", word)
elif re.match(replace_er_with, word):
    word = re.sub(replace_er_with, r"\1", word)
elif re.match(replace_ly_with, word):
    word = re.sub(replace_ly_with, r"\1", word)
elif re.match(replace_ment_with, word):
    word = re.sub(replace_ment_with, r"\1", word)
elif re.match(replace_able_with, word):
    word = re.sub(replace_able_with, r"\1", word)
elif re.match(replace_ible_with, word):
    word = re.sub(replace_ible_with, r"\1", word)
elif re.match(replace_ize_with, word):
    word = re.sub(replace_ize_with, r"\1", word)
elif re.match(replace_ion_with, word):
    word = re.sub(replace_ion_with, r"\1", word)

return word

```

```

stemmed_words_regex = [ porter_stem(word) for word in test_words ]
print(f"Original Word: {test_words}")
print(f"Stemmed Word: {stemmed_words_regex}")

```

```

Original Word: ['running', 'jumped', 'swimming', 'swimmer', 'apples', 'applesauce', 'programming', 'programmed', 'computers', 'computing', 'fa
Stemmed Word: ['runn', 'jump', 'swimm', 'swimm', 'apples', 'applesauce', 'programm', 'programm', 'computers', 'comput', 'fascinat', 'fascinat'

```

▼ Using nltk library

```
!pip install nltk
```

```

Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)

```

```

import nltk
from nltk.stem import PorterStemmer

# Download the Porter Stemmer data (you only need to do this once)
nltk.download('punkt')

```

```

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
True

```

```
stemmer = PorterStemmer()
```

```

stemmed_words_nltk = [ stemmer.stem(word) for word in test_words ]
print(f"Original Word: {test_words}")
print(f"Stemmed Word: {stemmed_words_regex}")

```

```

Original Word: ['running', 'jumped', 'swimming', 'swimmer', 'apples', 'applesauce', 'programming', 'programmed', 'computers', 'computing', 'fa
Stemmed Word: ['runn', 'jump', 'swimm', 'swimm', 'apples', 'applesauce', 'programm', 'programm', 'computers', 'comput', 'fascinat', 'fascinat'

```

▼ Comparison

```
import pandas as pd
df = pd.DataFrame([test_words, stemmed_words_regex, stemmed_words_nltk]).T
df
```

	0	1	2
0	running	runn	run
1	jumped	jump	jump
2	swimming	swimm	swim
3	swimmer	swimm	swimmer
4	apples	apples	appl
5	applesauce	applesauce	applesauc
6	programming	programm	program
7	programmed	programm	program
8	computers	computers	comput
9	computing	comput	comput
10	fascinating	fascinat	fascin
11	fascinated	fascinat	fascin
12	universality	universaliti	univers
13	universities	universiti	univers
14	difficulties	difficulti	difficulti
15	difficulty	difficulti	difficulti
16	categorize	categor	categor
17	categorization	categoriza	categor
18	categorizing	categoriz	categor
19	jumps	jumps	jump
20	laughing	laugh	laugh
21	laughter	laught	laughter
22	enjoyment	enjoy	enjoy
23	enjoyable	enjoy	enjoy
24	enjoyment	enjoy	enjoy