Porter stemmer -

1. Works differently for capital and small "Dying" and "dying" respectively in porter and not in Lancaster

Results:

print(lancaster.stem("Dying"))

print(porter.stem("Dying"))

dying

dy

print(lancaster.stem("dying"))

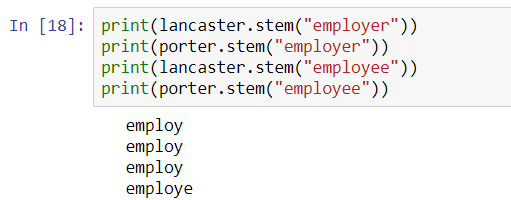
print(porter.stem("dying"))

dying

die

2. Last 2 letter vowels then -

1. changes to i if 2 different vowels at the end like "brownie" between lancaster and porter
2. changes to chopping off single letter at the end in words with same last letters(vowels) like "employee" and "proctee" between lancaster and porter



3. words like "cry" and "die" differ in both lancaster and porter like "die" is result for porter but in case of "cry" its "cri" and vice-versa. Many words which in main form ending with -y show results like "cry"

print(lancaster.stem("crying"))

print(porter.stem("crying"))

print(lancaster.stem("Dying"))

print(porter.stem("Dying"))

print(lancaster.stem("dying"))

print(porter.stem("dying"))

cry

cri

dying

dy

dying

die

4.-ible strictly chopped off the word in the results in lancaster stemmer and -ment in porter stemmer. Mostly suffix part is removed and results are in a rigid pattern of words with chopped off suffix, more in porter than in lancaster stemmer however lancaster returns base form successfully most times.



print(lancaster.stem("obliterate"))

print(porter.stem("obliterate"))

oblit

obliter

In [25]:



print(lancaster.stem("salutation"))

print(porter.stem("salutation"))

print(lancaster.stem("information"))

print(porter.stem("information"))

print(lancaster.stem("generation"))

print(porter.stem("generation"))

print(lancaster.stem("exclaimation"))

print(porter.stem("exclaimation"))

print(lancaster.stem("speculation"))

print(porter.stem("speculation"))

print(lancaster.stem("hibernation"))

print(porter.stem("hibernation"))

print(lancaster.stem("manipulation"))

print(porter.stem("manipulation"))

print(lancaster.stem("application"))

print(porter.stem("application"))

print(lancaster.stem("termination"))

print(porter.stem("termination"))

print(lancaster.stem("retribution"))

print(porter.stem("retribution"))

print(lancaster.stem("distribution"))

print(porter.stem("distribution"))

print(lancaster.stem("execution"))

print(porter.stem("execution"))

print(lancaster.stem("substitution"))

print(porter.stem("substitution"))

print(lancaster.stem("movement"))

print(porter.stem("movement"))

print(lancaster.stem("predicament"))

print(porter.stem("predicament"))

print(lancaster.stem("abolishment"))

print(porter.stem("abolishment"))

print(lancaster.stem("infringement"))

print(porter.stem("infringement"))

print(lancaster.stem("attainment"))

print(porter.stem("attainment"))

print(lancaster.stem("accomplishment"))

print(porter.stem("accomplishment"))

salut

salut

inform

inform

gen

gener

exclaim

exclaim

spec

specul

hibern

hibern

manip

manipul

apply

applic

termin

termin

retribut

retribut

distribut

distribut

execut

execut

substitut

substitut

mov

movement

predica

predica

abol

abolish

infr

infring

attain

attain

accompl

accomplish

print(lancaster.stem("incredible"))

print(porter.stem("incredible"))

print(lancaster.stem("invincible"))

print(porter.stem("invincible"))

print(lancaster.stem("forcible"))

print(porter.stem("forcible"))

print(lancaster.stem("visible"))

print(porter.stem("visible"))

incred

incred

invinc

invinc

forc

forcibl

vis

visibl

In [27]:



print(lancaster.stem("gillible"))

print(porter.stem("gullible"))

print(lancaster.stem("tangible"))

print(porter.stem("tangible"))

print(lancaster.stem("possible"))

print(porter.stem("possible"))

print(lancaster.stem("collapsible"))

print(porter.stem("collapsible"))

print(lancaster.stem("accessible"))

print(porter.stem("accessible"))

print(lancaster.stem("convertible"))

print(porter.stem("convertible"))

print(lancaster.stem("permissible"))

print(porter.stem("permissible"))

gill

gullibl

tang

tangibl

poss

possibl

collaps

collaps

access

access

convert

convert

permiss

permiss

In [31]:



print(lancaster.stem("abider"))

print(porter.stem("abider"))

print(lancaster.stem("abetter"))

print(porter.stem("abetter"))

print(lancaster.stem("abater"))

print(porter.stem("abater"))

print(lancaster.stem("adopter"))

print(porter.stem("adopter"))

print(lancaster.stem("adorner"))

print(porter.stem("adorner"))

abid

abid

abet

abett

ab

abat

adopt

adopt

adorn

adorn

In [32]:



print(lancaster.stem("ionization"))

print(porter.stem("ionization"))

print(lancaster.stem("activization"))

print(porter.stem("activization"))

print(lancaster.stem("realization"))

print(porter.stem("realization"))

ion

ioniz

act

activ

real

realiz

In [33]:



print(lancaster.stem("acidified"))

print(porter.stem("acidified"))

print(lancaster.stem("allied"))

print(porter.stem("allied"))

print(lancaster.stem("amplified"))

print(porter.stem("amplified"))

print(lancaster.stem("beautified"))

print(porter.stem("beautified"))

print(lancaster.stem("identified"))

print(porter.stem("identified"))

print(lancaster.stem("buried"))

print(porter.stem("buried"))

print(lancaster.stem("humidified"))

print(porter.stem("humidified"))

print(lancaster.stem("abducted"))

print(porter.stem("abducted"))

print(lancaster.stem("aborted"))

print(porter.stem("aborted"))

print(lancaster.stem("abandoned"))

print(porter.stem("abandoned"))

print(lancaster.stem("accorded"))

print(porter.stem("accorded"))

print(lancaster.stem("aligned"))

print(porter.stem("aligned"))

print(lancaster.stem("alleged"))

print(porter.stem("alleged"))

print(lancaster.stem("curated"))

print(porter.stem("curated"))

acid

acidifi

al

alli

ampl

amplifi

beaut

beautifi

ident

identifi

bury

buri

humid

humidifi

abduc

abduct

abort

abort

abandon

abandon

accord

accord

align

align

alleg

alleg

cur

curat