**Finding-FIRST-and-FOLLOW-of-given-grammar**

Code:-

import sys

sys.setrecursionlimit(60)

def first(string):

#print("first({})".format(string))

first\_ = set()

if string in non\_terminals:

alternatives = productions\_dict[string]

for alternative in alternatives:

first\_2 = first(alternative)

first\_ = first\_ |first\_2

elif string in terminals:

first\_ = {string}

elif string=='' or string=='@':

first\_ = {'@'}

else:

first\_2 = first(string[0])

if '@' in first\_2:

i = 1

while '@' in first\_2:

#print("inside while")

first\_ = first\_ | (first\_2 - {'@'})

#print('string[i:]=', string[i:])

if string[i:] in terminals:

first\_ = first\_ | {string[i:]}

break

elif string[i:] == '':

first\_ = first\_ | {'@'}

break

first\_2 = first(string[i:])

first\_ = first\_ | first\_2 - {'@'}

i += 1

else:

first\_ = first\_ | first\_2

#print("returning for first({})".format(string),first\_)

return first\_

def follow(nT):

#print("inside follow({})".format(nT))

follow\_ = set()

#print("FOLLOW", FOLLOW)

prods = productions\_dict.items()

if nT==starting\_symbol:

follow\_ = follow\_ | {'$'}

for nt,rhs in prods:

#print("nt to rhs", nt,rhs)

for alt in rhs:

for char in alt:

if char==nT:

following\_str = alt[alt.index(char) + 1:]

if following\_str=='':

if nt==nT:

continue

else:

follow\_ = follow\_ | follow(nt)

else:

follow\_2 = first(following\_str)

if '@' in follow\_2:

follow\_ = follow\_ | follow\_2-{'@'}

follow\_ = follow\_ | follow(nt)

else:

follow\_ = follow\_ | follow\_2

#print("returning for follow({})".format(nT),follow\_)

return follow\_

no\_of\_terminals=int(input("Enter no. of terminals: "))

terminals = []

print("Enter the terminals :")

for \_ in range(no\_of\_terminals):

terminals.append(input())

no\_of\_non\_terminals=int(input("Enter no. of non terminals: "))

non\_terminals = []

print("Enter the non terminals :")

for \_ in range(no\_of\_non\_terminals):

non\_terminals.append(input())

starting\_symbol = input("Enter the starting symbol: ")

no\_of\_productions = int(input("Enter no of productions: "))

productions = []

print("Enter the productions:")

for \_ in range(no\_of\_productions):

productions.append(input())

#print("terminals", terminals)

#print("non terminals", non\_terminals)

#print("productions",productions)

productions\_dict = {}

for nT in non\_terminals:

productions\_dict[nT] = []

#print("productions\_dict",productions\_dict)

for production in productions:

nonterm\_to\_prod = production.split("->")

alternatives = nonterm\_to\_prod[1].split("/")

for alternative in alternatives:

productions\_dict[nonterm\_to\_prod[0]].append(alternative)

#print("productions\_dict",productions\_dict)

#print("nonterm\_to\_prod",nonterm\_to\_prod)

#print("alternatives",alternatives)

FIRST = {}

FOLLOW = {}

for non\_terminal in non\_terminals:

FIRST[non\_terminal] = set()

for non\_terminal in non\_terminals:

FOLLOW[non\_terminal] = set()

#print("FIRST",FIRST)

for non\_terminal in non\_terminals:

FIRST[non\_terminal] = FIRST[non\_terminal] | first(non\_terminal)

#print("FIRST",FIRST)

FOLLOW[starting\_symbol] = FOLLOW[starting\_symbol] | {'$'}

for non\_terminal in non\_terminals:

FOLLOW[non\_terminal] = FOLLOW[non\_terminal] | follow(non\_terminal)

#print("FOLLOW", FOLLOW)

print("{: ^20}{: ^20}{: ^20}".format('Non Terminals','First','Follow'))

for non\_terminal in non\_terminals:

print("{: ^20}{: ^20}{: ^20}".format(non\_terminal,str(FIRST[non\_terminal]),str(FOLLOW[non\_terminal])))

Output:-

