

17-Mar-22

## Experiment 8 - Computation of leading and trailing

Dhawal Patil

RA1911003010575

CSE A2

### Aim:

To write a program to show Computation of leading and trailing using Python language.

### Algorithm:

1. For Leading, check for the first non-terminal.
2. If found, print it.
3. Look for next production for the same non-terminal.
4. If not found, recursively call the procedure for the single non-terminal present before the comma or End Of Production String.
5. Include it's results in the result of this non-terminal.
6. For trailing, we compute same as leading but we start from the end of the production to the beginning.
7. Stop

### Code:

```
a = ["E=E+T", "E=T", "T=T*F", "T=F", "F=(E)", "F=i"]
```

```
rules = {}
terms = []
for i in a:
    temp = i.split("=")
    terms.append(temp[0])
    try:
        rules[temp[0]] += [temp[1]]
    except:
        rules[temp[0]] = [temp[1]]
```

```
terms = list(set(terms))
print(rules, terms)
```

```
def leading(gram, rules, term, start):
    s = []
    if gram[0] not in terms:
        return gram[0]
    elif len(gram) == 1:
        return [0]
    elif gram[1] not in terms and gram[-1] is not start:
        for i in rules[gram[-1]]:
            s += leading(i, rules, gram[-1], start)
        s += [gram[1]]
    return s
```

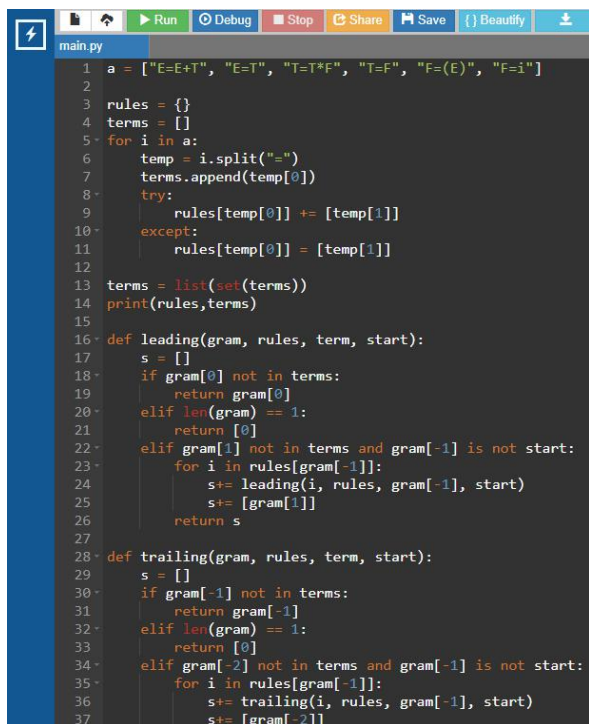
```
def trailing(gram, rules, term, start):
```

```

s = []
if gram[-1] not in terms:
    return gram[-1]
elif len(gram) == 1:
    return [0]
elif gram[-2] not in terms and gram[-1] is not start:
    for i in rules[gram[-1]]:
        s+= trailing(i, rules, gram[-1], start)
        s+= [gram[-2]]
    return s

leads = {}
trails = {}
for i in terms:
    s = [0]
    for j in rules[i]:
        s+=leading(j,rules,i,i)
    s = set(s)
    s.remove(0)
    leads[i] = s
    s = [0]
    for j in rules[i]:
        s+=trailing(j,rules,i,i)
    s = set(s)
    s.remove(0)
    trails[i] = s
for i in terms:
    print("LEADING("+i+":",leads[i])
for i in terms:
    print("TRAILING("+i+":",trails[i])

```



```

main.py
1  a = ["E=E+T", "E=T", "T=T*F", "T=F", "F=(E)", "F=i"]
2
3  rules = {}
4  terms = []
5  for i in a:
6      temp = i.split("=")
7      terms.append(temp[0])
8      try:
9          rules[temp[0]] += [temp[1]]
10     except:
11         rules[temp[0]] = [temp[1]]
12
13 terms = list(set(terms))
14 print(rules,terms)
15
16 def leading(gram, rules, term, start):
17     s = []
18     if gram[0] not in terms:
19         return gram[0]
20     elif len(gram) == 1:
21         return [0]
22     elif gram[1] not in terms and gram[-1] is not start:
23         for i in rules[gram[-1]]:
24             s+= leading(i, rules, gram[-1], start)
25             s+= [gram[1]]
26     return s
27
28 def trailing(gram, rules, term, start):
29     s = []
30     if gram[-1] not in terms:
31         return gram[-1]
32     elif len(gram) == 1:
33         return [0]
34     elif gram[-2] not in terms and gram[-1] is not start:
35         for i in rules[gram[-1]]:
36             s+= trailing(i, rules, gram[-1], start)
37             s+= [gram[-2]]

```

```

38         return s
39
40 leads = {}
41 trails = {}
42 for i in terms:
43     s = [0]
44     for j in rules[i]:
45         s+=leading(j,rules,i,i)
46     s = set(s)
47     s.remove(0)
48     leads[i] = s
49     s = [0]
50     for j in rules[i]:
51         s+=trailing(j,rules,i,i)
52     s = set(s)
53     s.remove(0)
54     trails[i] = s
55 for i in terms:
56     print("LEADING("+i+":",leads[i])
57 for i in terms:
58     print("TRAILING("+i+":",trails[i])

```

## Output:

```

{'E': ['E+T', 'T'], 'T': ['T*F', 'F'], 'F': ['(E)', 'i']} ['E', 'T', 'F']
LEADING(E): {'*', '(', 'i', '+'}
LEADING(T): {'i', '(', '*'}
LEADING(F): {'i', '('}
TRAILING(E): {'*', 'i', '+', ')'}
TRAILING(T): {'i', '*', ')'}
TRAILING(F): {'i', ')'}

...Program finished with exit code 0
Press ENTER to exit console.

```

## Result:

A program for Computation of leading and trailing was run successfully.