

# INFIX TO POSTFIX,PREFIX

NAME : ATHRESH KUMAR LABDE

RA1911033010146

M1

**AIM:** To verify INFIX to POSTFIX,PREFIX by implementing it in the code.

## **ALGORITHM:**

**Step 1:** If the scanned character is an operand, put it into postfix expression.

**Step 2:** If the scanned character is an operator and operator's stack is empty, push the operator into operators' stack.

**Step 3:** If the operator's stack is not empty, there may be following possibilities.

If the precedence of scanned operator is greater than the top most operator of operator's stack, push this operator into operator 's stack.

If the precedence of scanned operator is less than the top most operator of operator's stack, pop the operators from operator's stack until we find a low precedence operator than the scanned character.

If the precedence of scanned operator is equal then check the associativity of the operator. If associativity left to right then pop the operators from stack until we find a low precedence operator. If associativity right to left then simply put into stack.

If the scanned character is opening round bracket ( '(' ), push it into operator's stack.

If the scanned character is closing round bracket ( ')' ), pop out operators from operator's stack until we find an opening bracket '(' ).

Repeat Step 1,2 and 3 till expression has character

**Step 4:** Now pop out all the remaining operators from the operator's stack and push into postfix expression.

**Step 5:** Exit.

## **CODE :**

```
OPERATORS = set(['+', '-', '*', '/', '(', ')'])
```

```
PRI = {'+': 1, '-': 1, '*': 2, '/': 2}
```

```
### INFIX ==> POSTFIX ###
```

```
def infix_to_postfix(formula):
```

```
stack = [] # only pop when the coming op has priority
```

```
output = ""
```

```
for ch in formula:
```

```
    if ch not in OPERATORS:
```

```
        output += ch
```

```
    elif ch == '(':
```

```
        stack.append('(')
```

```
    elif ch == ')':
```

```
        while stack and stack[-1] != '(':
```

```
            output += stack.pop()
```

```
        stack.pop() # pop '('
```

```
    else:
```

```
        while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
```

```
            output += stack.pop()
```

```
        stack.append(ch)
```

```
    # leftover
```

```
while stack:
```

```
    output += stack.pop()
```

```
print(f'POSTFIX: {output}')
```

```
return output
```

```
### INFIX ==> PREFIX ###
```

```
def infix_to_prefix(formula):
```

```
    op_stack = []
```

```
    exp_stack = []
```

```
    for ch in formula:
```

```
        if not ch in OPERATORS:
```

```
            exp_stack.append(ch)
```

```
        elif ch == '(':
```

```
            op_stack.append(ch)
```

```
        elif ch == ')':
```

```
            while op_stack[-1] != '(':
```

```
                op = op_stack.pop()
```

```
                a = exp_stack.pop()
```

```
                b = exp_stack.pop()
```

```
                exp_stack.append(op + b + a)
```

```
            op_stack.pop() # pop '('
```

```
        else:
```

```
            while op_stack and op_stack[-1] != '(' and PRI[ch] <= PRI[op_stack[-1]]:
```

```
                op = op_stack.pop()
```

```
                a = exp_stack.pop()
```

```
        b = exp_stack.pop()

        exp_stack.append(op + b + a)

    op_stack.append(ch)

    # leftover

while op_stack:

    op = op_stack.pop()

    a = exp_stack.pop()

    b = exp_stack.pop()

    exp_stack.append(op + b + a)

print(f'PREFIX: {exp_stack[-1]}')

return exp_stack[-1]

expres = input("INPUT THE EXPRESSION: ")

pre = infix_to_prefix(expres)

pos = infix_to_postfix(expres)
```

The screenshot displays a web browser window with the URL `programiz.com/python-programming/online-compiler/`. The page header includes the Programiz logo and a "Learn Python App" button. The main interface is divided into two sections: a code editor on the left and a shell output window on the right.

The code editor contains a Python script named `main.py` with the following code:

```
85
86     exp_stack.append(op + b + a)
87
88     op_stack.append(ch)
89
90     # leftover
91
92 while op_stack:
93     op = op_stack.pop()
94
95     a = exp_stack.pop()
96
97     b = exp_stack.pop()
98
99     exp_stack.append(op + b + a)
100
101 print(f'PREFIX: {exp_stack[-1]}')
102
103 return exp_stack[-1]
104
105 expres = input("INPUT THE EXPRESSION: ")
106
107 pre = infix_to_prefix(expres)
108
```

The shell output window shows the following text:

```
INPUT THE EXPRESSION: A*B^C/R
PREFIX: +^/CR
POSTFIX: AB^CR/+
> |
```

A promotional banner for "Introducing the most interactive Python Course" is visible in the bottom right corner of the code editor area. The Windows taskbar at the bottom shows the time as 10:16 on 02-04-2022.

**RESULT :** Hence the result is verified and implemented in form of the code.