COMPUTER SCIENCE AND DA

Data Structures through Python

Stack

Lecture No. 05

By- Kashif Sir

Topics to be covered

Last Class

↳ Stack Implementation

- ↳ Brute Force Approach
- ↳ List
- ↳ Deque
- ↳ numpy
- ↳ LifoQueue

→ Application of Stack

↳ Paranthesis Balancing.

Agenda

Expression Conversion

EXPRESSION CONVERSION

Expression : Any executable statement

There are different types of expressions:

1) Input Expression — input( )
2) Output Expression — print( )
3) Declaration / Initialization — a = 3, b = "Hello"
4) Computation / Arithmetic / Assignment — a = b + 3, c = print("hello")
5) Definition — def: fun( )
6) Conditional — if - elif - else
7) Iterative — for, while

HLL → (interpreter) → LLL

class A:

EXPRESSION CONVERSION

3 + 2 * 4 / 5 ⇒ 3 + 8 / 5

7

3 + 1.6

= 4.6

* / +

If user wants the order of evaluation to be +, *, /

(((3 + 2) * 4) / 5)

13

EXPRESSION CONVERSION

There are 3 different ways to write expression:

⇒ 1) Infix operand1 operator operand2 a+b

2) Prefix operator operand1 operand2 +ab

3) Postfix Operand1 operand2 operator ab+

Infix to Postfix Conversion

Let X be the Infix and Y be the Postfix and S is a empty stack.

1) Scan element of X one by one from left to right.

2) If element == operand then add it to Y

3) If element == (then push it to S

4) If element == operator(O_s), then check top of the stack
   - If stack is empty, push O_s
   - If Top of stack is (then push O_s
   - If Top of stack is O_T: if (O_T >= O_s) pop O_T and add to Y else push O_s to stack.

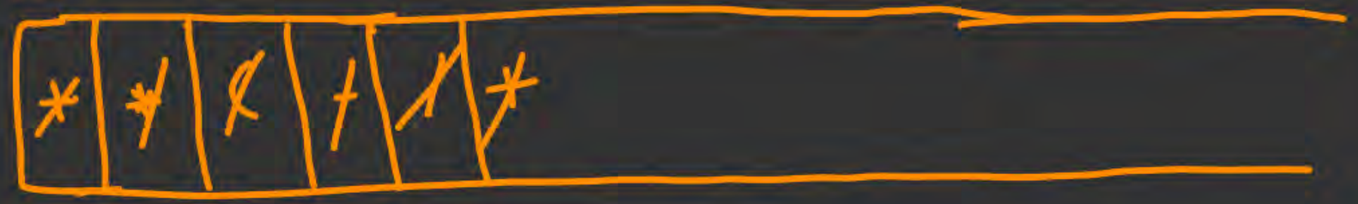
5) If element ==), pop elements from S and add to Y, until first (

6) Repeat 2, 3, 4, 5 for all elements until expression is completed. After that empty stack by popping and adding to Y.

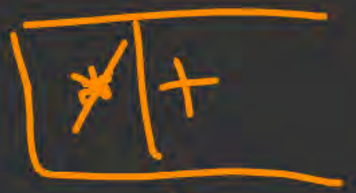
7) Y is the postfix expression now.

$X = A + B * (C - D / E) + F$

S



$Y = ABCDE/- * + F +$



$O_T = *$

$O_S = +$

$Y = *$

$X = (a * b) - (c / (d \wedge e) - f) * (g \wedge h)$

$Y = ab * cde \wedge / f - gh \wedge * -$

THANK - YOU