CMPSC-132: Programming and Computation II
Fall 2018

Homework 4

Due Date: 11/16/2018, 11:59PM
100 pts

Instructions:

- The work in this assignment must be completed alone.
- Use the starter code provided on this CANVAS assignment. Do not change the function names.
- The file name must be HW4.py (incorrect name files will get a -10 point deduction)
- When any function returns an error, it must be a string containing "error"
- A doctest is provided as an example of code functionality. Getting the same result as the doctest does not guarantee full credit. You are responsible for debugging and testing your code with enough data.
- Do not include test code outside any function in the upload. Printing unwanted or ill-formatted data to output will cause the test cases to fail. Remove all your testing code before uploading your file. Do not include the input() function in your submission.

Goal:

Modify the function *calculator*(*expr*) in HW3.py so that it supports parentheses in *expr*. An example of a valid expression is:

"2 + 3 * (-2 +(-3) *(5^2 - 2*3^(-2)) *(-4)) * (2 /8 + 2*(3 – 1/ 3)) - 2/ 3^2"

Notes:

- Copy your code from *calculator*(*expr*) into the function *_calculator*(*expr*). Your *calculator*(*expr*) in HW3.py must call *_calculator*(*expr*)
- Modify getNextNumber to allow negation in expressions like this: "4+2*-4+1"
- Add the stack code from Lab 9 or you queue code from Lab 10 into your HW4 script. Your *calculator*(*expr*) function must use the stack or the queue for parenthesis matching, otherwise, no credit will be given

Function requirements:

- ✓ The function must **return** the computed value if *expr* is a correct formula, otherwise it must return an error message.
- ✓ When any function returns a numeric value, it must be float
- ✓ Do not use *exec* or *eval* function. You will not receive credit if your program uses any of the two functions anywhere

Grading Notes:

- The grading script will feed 5 randomly chosen test inputs, each for 20 points. One of them will be an input that should cause an error such as "(4 * 5/ (2 + (5 ^2))", whose expected returned value is an error message.

```
Examples:
>>> calculator("3*(10 - 2*3)")
12.0
>>> calculator(" -2 / (- 4) * (3  - 2*( 4- 2^3)) + 3")
8.5
>>> calculator("2*(4+2*(5-3^2)+1)+4")
-2.0
>>> calculator("-(-2)*10 - 3*(2 - 3*2) ")
32.0
>>> calculator("-(-2)*10 - 3*(2 - 3*2)) ")
error
>>> calculator("-(-2)*10 - 3*/(2 - 3*2) ")
error
```

Deliverables:

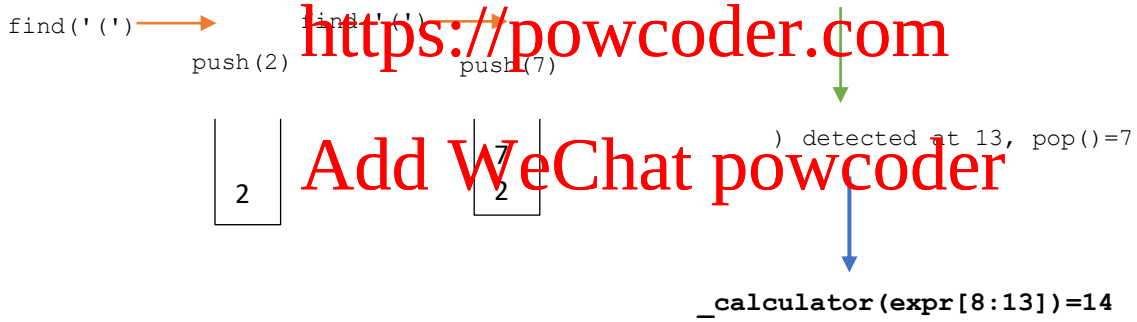
- Include all the functions in your script named HW4.py. Submit it to the HW4 CANVAS assignment before the due date

Tips for using the Stack:

Let's consider and expression with no negation:

expr="2*(4+2*(5+3^2)+1)+4"

pos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	2	*	(	4	+	2	*	(	5	+	3	^	2	)	+	1	)	+	4

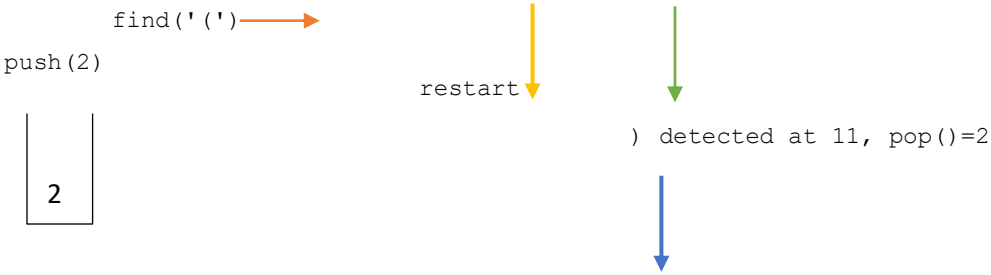


_calculator(expr[8:13])=14

We can use the stack to find an innermost pair of parentheses and the use _calculator to compute the value of the inside expression. Then, replace the parentheses and its inside by the calculated result. Pop from the stack, and continue:

expr="2*(4+2*14+1)+4"

pos	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	2	*	(	4	+	2	*	1	4	+	1	)	+	4



_calculator(expr[3:11])=33

Replace it by 33. Now expr = “2*33+4”. Call _calculator(expr) to find the result, 70

Now, let’s consider and expression with negation inside the parentheses:

expr=”2*(4+2*(5-3^2)+1)+4”

pos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	2	*	(	4	+	2	*	(	5	-	3	^	2	)	+	1	)	+	4

-4

4+2*-4+1

_calculator('4+2*-4+1') must return -3

We need to modify getNextNumber to allow such operation

pos	0	1	2	3	4	5	6	7
	4	+	2	*	-	4	+	1

_calculator('4+2*-4+1'):

- pos=4 right after detecting *
- _calculator calls getNextNumber(expr, pos) #expr[4:]
- getNextNumber should disregard the - at the 0th position when calling findNextOpr, so that – is not part of the expression and we get the position of + instead. But we must leave it as part of newNumber
-

pos	0	1	2	3	4	5	6	7
	4	+	2	*	-	4	+	1

expr[pos:]

getNextNumber('4+2*-4+1',4)

newNumber

newOpr

- 4 + 1

pass to findNextOpr