Reverse Polish Notation (RPN) Calculator

Christopher Norman

Introduction

RPN is a form of mathematical notation where operators follow their operands. For example infix notation would be:

$$(9 * 2) * 3 - 4$$

Where as in RPN (postfix) it is written as:

$$9 \quad 2 \; * \; 3 \; * \; 4 \; -$$

Postfix notation allows the expression to be read and evaluated from left to right for any mathematical expression. It also means that a computer can parse the expression more efficiently. In this repo an RPN calculator is implemented as a Python object. To use it create a new `RPN()` object and call the `evaluate()` function on this object with an expression as a string as the argument. The output is accessible through the `result` attribute of the object.

1 How would you implement an infix notation calculator on top of your RPN calculator?

To implement an infix notation calculator it would make sense to add a new function `infix_to_postfix(str)` that takes an input string of an ordinary arithmetic expression such as `"(2+8)/2"` and converts it to postfix/RPN notation, in this case `"2 8 + 2 /"` which can then be passed to the existing function `evaluate()` and calculated. The result can be retrieved as normal through the `result` attribute of the `RPN` object.

2 How would you deploy your calculator as a service in a cloud environment?

One option for using this calculator as a microservice in a cloud environment is adding it to a *Flask* or other web application. This application would allow users to send an API request with an RPN expression to be evaluated. The application could then be hosted on a cloud server. This application can then be containerised with *Docker* and then deployed using a cloud service such as *AWS* or *Azure.*

Furthermore, in a cloud environment it may make more sense to use custom exceptions and catch them at a higher level as currently, errors are caught within the program and the user is notified through print statements in the command line. It may also be better to have the function `evaluate()` return the calculated output instead of storing it as a class variable.