Programming Techniques – Project #2:

The Not-So-Basic Calculator

Introduction

Tired of using your laptop's calculator? Develop your own! Fancier, more flexible, more reliable (no wonder, *you* developed it), it will evaluate the most complex expressions you can think of… for free!

The project must be prepared in pairs, defended on Thursday January 11th and submitted on Campus the same day by 23:55.

Please read this paper thoroughly before you begin the project. If you have any question, please post it in the Q&A forum, in the "Project #2" thread.

Contents

1 Schedule

The schedule of the project is as follows:

Date	Step
December 8th	Release of the project's paper
December 14th	Follow-up session
January 11th	Defense of the project
January 11th 23:55	Submission of the project on Campus

2 Requirements

The calculator is interactive. It displays a prompt, e.g. ?, and waits for the user to enter an expression. When the user hits the enter key, the calculator evaluates the expression, prints the result and displays the prompt again. This process repeats until the user hits the enter key twice without entering any expression, which causes the calculator to exit.

The calculator evaluates expressions of type integer, string and boolean. It also enables the user to define variables, e.g. to store temporary results, and use these variables in expressions.

To help you with the development of the calculator, we suggest the following steps.

2.1 Expressions

Expressions are of any length. They can include parenthesized sub-expressions, with arbitrary nesting levels. Operands, operators and parenthesis are delimited by any number (including 0) of space characters.

2.1.1 Integer Expressions

Integer expressions include integer operands and the 4 basic integer operators: `+ - x /`. Standard associativity and precedence rules apply.

As a reminder, all operators are left-associative. Thus, `1 + 2 - 3` is evaluated as `(1 + 2) - 3`. The precedence of the operators is given in the table below. Thus, `1 + 2 x 3` is evaluated as `1 + (2 x 3)`.

Precedence	Operators
1	x /
2	+ -

The example below shows a calculator session with integer expressions. Items in red are printed by the calculator, items in black are input by the user.

```
? 1
1
? (1)
1
? 1 - 2
-1
```

```
? 4 / 2 x 3
6
? 4 x 2 / 3
2
? 1 + 2 x 3
7
? (1 + 2) x 3
9
? -1 + -1 + (-1 - -1)
-2
?
```

2.1.2 String Expressions

String expressions include string operands and the string concatenation operator: +. A string is a sequence of characters of any length (including 0) enclosed between double quotes. The + operator is left associative.

The example below shows a calculator session with string expressions.

```
? "hello"
hello
? "hello" + "" + "world"
helloworld
? "hello" + " " + ("world" + "!")
hello world!
?
```

2.1.3 Boolean Expressions

Boolean expressions include boolean literals: `true` and `false`, comparison operators: `==` `<>` `<` `<=` `>` `>=`, and the 3 basic boolean operators: `or` `and` `not`. All operators are left-associative. The table below gives the precedence of the operators that can be involved in a boolean expression.

Precedence	Operators
1	x /
2	+ –
3	== <> < <= > >=
4	not
5	and
6	or

The example below shows a calculator session with boolean expressions.

```
? true
true
```

```
? true or false and false
true
? (true or false) and false
false
? not false and not false
true
? 1 < 0
false
? "Hi" <> "Hello"
true
? "Hi" < "Hello"
false
? 1 + 2 + 3 = 2 x 3
true
? 1 < 5 and 5 < 10
true
? 1 < 5 and not "Hi" == "Hello"
true
?
```

2.1.4 Promoting to String Type

As a convenience, the calculator automatically converts integer and boolean values to a string when they are concatenated to another string. Thus, the following are valid string expressions:

```
? "1 + 2 = " + 1 + 2
1 + 2 = 12
? "1 + 2 = " + (1 + 2)
1 + 2 = 3
? "1 + 2 > 3 = " + (1 + 2 > 3)
1 + 2 > 3 = false
? "temperature: " + 5 + "°C, cold: " + (5 < 25)
temperature: 5°C, cold: true
?
```

2.2 Variables

The calculator allows the user to define variables of all types, for example to store temporary results. A variable name is a sequence of alphanumeric characters beginning with an alphabetical character. String names are of any length and case-sensitive.

Variable management includes variable assignment and variable substitution, as detailed below.

2.2.1 Variable Assignment

A variable can be assigned the value of an expression of any type. The syntax of an assignment statement is as follows:

variableName = expression

The calculator does not display anything when executing such statements. Here are some examples of variable assignments:

```
? x = 1
? y = 1 + 2
? message = "hello"
? errorCause = "type" + " mismatch"
? oneIsOdd = 1 % 0 == 1
? areDistinctColors = "blue" <> "orange"
?
```

2.2.2 Variable Substitution

A variable can be used as an operand in any expression. The calculator replaces the variable with its current value before evaluating the expression. Here are some examples of expressions and assignment statements using the variables defined in the previous paragraph:

```
? x
1
? x + 2 x y
7
? "error: " + errorCause + "!"
error: type mismatch
? oneIsOdd and not areDistinctColors
false
? x = x + y
? x
4
? message = "one is odd: " + oneIsOdd
? message
one is odd: true
?
```

2.3 Error Handling

The calculator displays an error message when the input expression is syntactically or semantically incorrect. Below are some error message examples.

```
? 1 +
error: missing operand (near '+')
? (1 + 2) x 3)
error: unmatched parenthesis (')')
? "temperature: " + temp
error: unknown variable ('temp')
```

```
? false + 1
error: type mismatch (boolean + integer)
? 1 + "hello”
error: type mismatch (integer + string)
?
```

3 Bonus Features

Below is a non-exhaustive list of additional features you can implement.

Configuration File: You can redefine operators, boolean literals, parenthesis, the prompt, etc. in a configuration file, so that the calculator will accept expressions as follows:

```
-> ]1 plus 2[ times 3
9
-> wrong . !right
wrong
```

Additional Operators: You can implement additional operators, especially string operators, so that the calculator will accept expressions as follows:

```
? "hello world!" - "l"
heo word!
? #"hello"
olleh
```

Polish Notation: You can make your calculator accept expressions in Polish Notation, as follows:

```
? 1 2 + 3 x
9
?
```

Graphical User Interface: You can add a graphical user interface to your calculator, using Python’s tkinter module. Here is a tutorial on using this module:
https://www.tutorialspoint.com/python3/python_gui_programming.htm

4 Marking Scheme

The (tentative) marking scheme is as follows:

Item	**Marks**
Code	10
integer expressions	*2*
string expressions	*1*
boolean expressions	*3*
variables	*2*
code quality	*2*

Item	Marks
Defense	5
Report	5
Total	**20**

Code quality: The code must be commented. It must comply with the standard naming rules, especially for variables and functions. A special attention will be paid to the architecture of your code: decomposition into modules, function prototypes, etc.

Report: The report is between 4 to 6 pages long. It first describes the architecture of the calculator (i.e. the code modules it contains and their purpose), its main data structures and main algorithms. Next, it describes your achievements with respect to the requirements and the problems you faced. In appendix, the report gives the screenshots of the execution of each of the examples given above.

5 Deliverable

The deliverable consists of a single archive named LASTNAME1.FirstName1.LASTNAME2.FirstName2.rar or .zip to identify the project members. The archive contains:

- the python files, including the main file: main.py
- the report in pdf format: report.pdf

The deliverable must be submitted on Campus on Thursday January 11th by 23:55.

6 Project Defense

Each project team has 20 minutes to defend its work, as follows:

- During the first 10 minutes, you will briefly introduce the project, describe the organization of the team, expose your achievements with respect to the requirements, and the problems you faced. *Both team members must speak in turn*. Note that you must *prepare* and *rehearse* your speech.

- For the next 10 minutes, you will answer the questions of the examiner. *Each team member must be able to answer any question, on any part of the project.*

Please note that, depending on the quality of their performance, project members may get different marks.

Have serious fun!