University of St. Gallen

School of Computer Science

Fundamentals of Computer Science (3,232), HS2020
S. Mayer, B. Weber, R. Seiger

Assignment 1: Programming Basics

Deadline: Sep 24, 2020; 12:00 CEST

Introduction As part of this assignment, you will create and implement a computer program from scratch. The tasks are based on the guided exercise session of Sep 14, 2020. Together with last week's introductory assignment[1], that session contains all information required to solve this assignment.

Prerequisites To solve this assignment, you require a computer with a Windows, Mac, or Linux based operating system, installed Python and Visual Studio Code. VS Code with Live Share extension is this lecture's supported programming environment.

① A Simple Calculator

Implement a simple calculator (*interactive-calculator.py*), as demonstrated during the first guided exercise.[2] Your calculator should provide the four basic arithmetic operations and adhere to this specification:

- The program shall ask the user to enter/select the desired operation (*add/subtract/multiply/divide/end*).
- When the user selects an arithmetic operation, the program shall ask the user to input the two operands, calculate the result, and print the result.
- The program is executed repeatedly until the user actively stops it using the command *end*.

Fig. 1 on the next page displays example outputs from executions of the program in the terminal or within Visual Studio Code.

Hint: Here you will have to find a way of converting the user inputs, which are Strings, into Integers such that you can apply mathematical operations to them.

② History-Extension for your Calculator

Extend your program with a 'history': The program shall record each executed arithmetic operation using a *Python List*. Furthermore, it should be possible to display this list using an additional command, *history*. Entries in the history should be Strings formatted in the following way. Examples are shown in Fig. 1.

(position) operation operand1 operand2 -> result

Hint: Try to use *f-Strings*[3] to correctly format an entry of the history list.

[1] https://learning.unisg.ch/courses/7522/assignments/17122

[2] Note to Windows users: To simplify file handling, activate the option *Show File Extensions* in Windows Explorer. This resource describes how this is done: https://www.thewindowsclub.com/show-file-extensions-in-windows

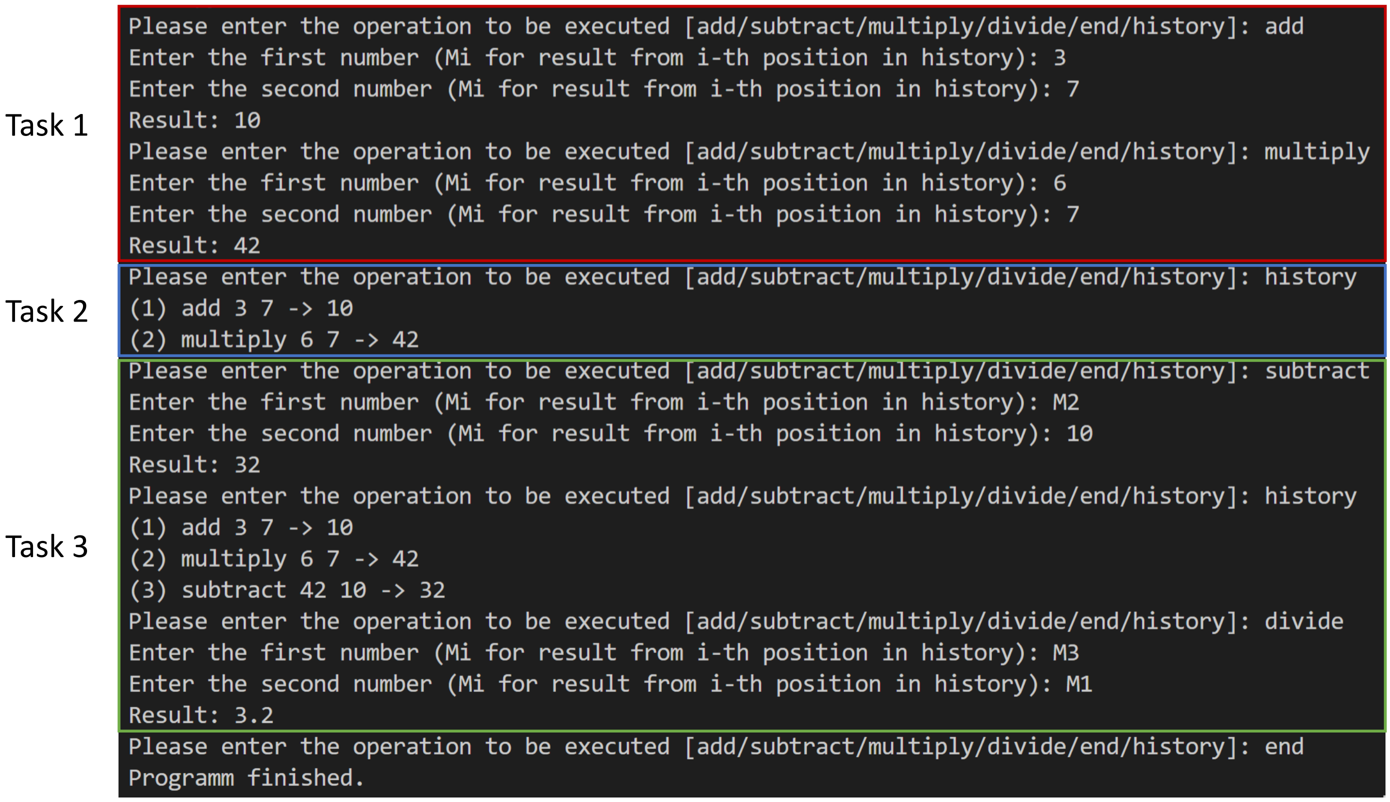
[3] https://realpython.com/python-f-strings/#f-strings-a-new-and-improved-way-to-format-strings-in-python

3 **Recall-Extension for your Calculator** (Advanced)

Based on the history extension built in Task 2, extend your program with a recall function for a result of a previous calculation to be used in your current calculation. This means that you should be able to use the result of a calculation from the history as input for the first or second operand of the current calculation. This is specified by the user via `Mi` (with `i` being the i-th result of the current history) as one of the operands. That is, When a user enters `(multiply; 2; 3)` the calculator shall perform this operation as specified; if the user subsequently enters `(subtract; M1; 5)` the calculator shall *recall* the result of the first operation (i.e., the multiply) and then perform the operation and store it in history; next, `(add; M1; M2)` would add the results of the previous operations. The example output in Fig. 1 will make this more clear.

Note that just like regular operations, operations with recalled arguments should be kept in the history.

Hint: Here you should take a closer look at working with Strings[4] in Python. You may need to find the position of certain characters or substrings with in a String and extract substrings from a certain position to a certain position within a string.



```
Please enter the operation to be executed [add/subtract/multiply/divide/end/history]: add
Enter the first number (Mi for result from i-th position in history): 3
Enter the second number (Mi for result from i-th position in history): 7
Result: 10
Please enter the operation to be executed [add/subtract/multiply/divide/end/history]: multiply
Enter the first number (Mi for result from i-th position in history): 6
Enter the second number (Mi for result from i-th position in history): 7
Result: 42
Please enter the operation to be executed [add/subtract/multiply/divide/end/history]: history
(1) add 3 7 -> 10
(2) multiply 6 7 -> 42
Please enter the operation to be executed [add/subtract/multiply/divide/end/history]: subtract
Enter the first number (Mi for result from i-th position in history): M2
Enter the second number (Mi for result from i-th position in history): 10
Result: 32
Please enter the operation to be executed [add/subtract/multiply/divide/end/history]: history
(1) add 3 7 -> 10
(2) multiply 6 7 -> 42
(3) subtract 42 10 -> 32
Please enter the operation to be executed [add/subtract/multiply/divide/end/history]: divide
Enter the first number (Mi for result from i-th position in history): M3
Enter the second number (Mi for result from i-th position in history): M1
Result: 3.2
Please enter the operation to be executed [add/subtract/multiply/divide/end/history]: end
Programm finished.
```

Figure 1: Example output after tasks 1, 2 and 3.

4 **Error Handling** (Optional)

Think about possible errors that may occur when using the calculator and try to prevent or handle them, e. g, division by zero, entering non-valid operations and operands, accessing from history that does not exist, etc.

Hint: Here it may already be useful to have a look at exceptions[5] and execption handling[6] in Python. You will encounter these, e.g., when trying to convert a String into an int (*ValueError*) and when accessing an index of a list that does not exist (*IndexError*). We will use these mechanisms in Assignment 2.

[4] https://www.programiz.com/python-programming/string

[5] https://www.programiz.com/python-programming/exceptions

[6] https://www.programiz.com/python-programming/exception-handling

Hand-in Instructions Create a `zip` file including the *interactive-calculator.py* Python file. Make sure the Python file contains your ID and name at its beginning. To receive feedback from your tutor, submit the zip file by the deadline indicated on top of this sheet via the course page for Assignment 1 on Canvas[7].

[7] https://learning.unisg.ch/courses/7522/assignments