**Course: Introduction to Data Science (DS2006) - Laboratory 05**

**Student: Philip Haglund**

Over the last few classes, we have seen variables, how to print things to the screen, how to get input from the user, how to make choices using selection structures (if-elif-else), how to make loops (using for and while), how to use lists and how to save files. Today we have some coding tasks that will need you to use all of this knowledge together.

* **Task 1**: Look at the code shown in Figure 1. What is the purpose of this code?

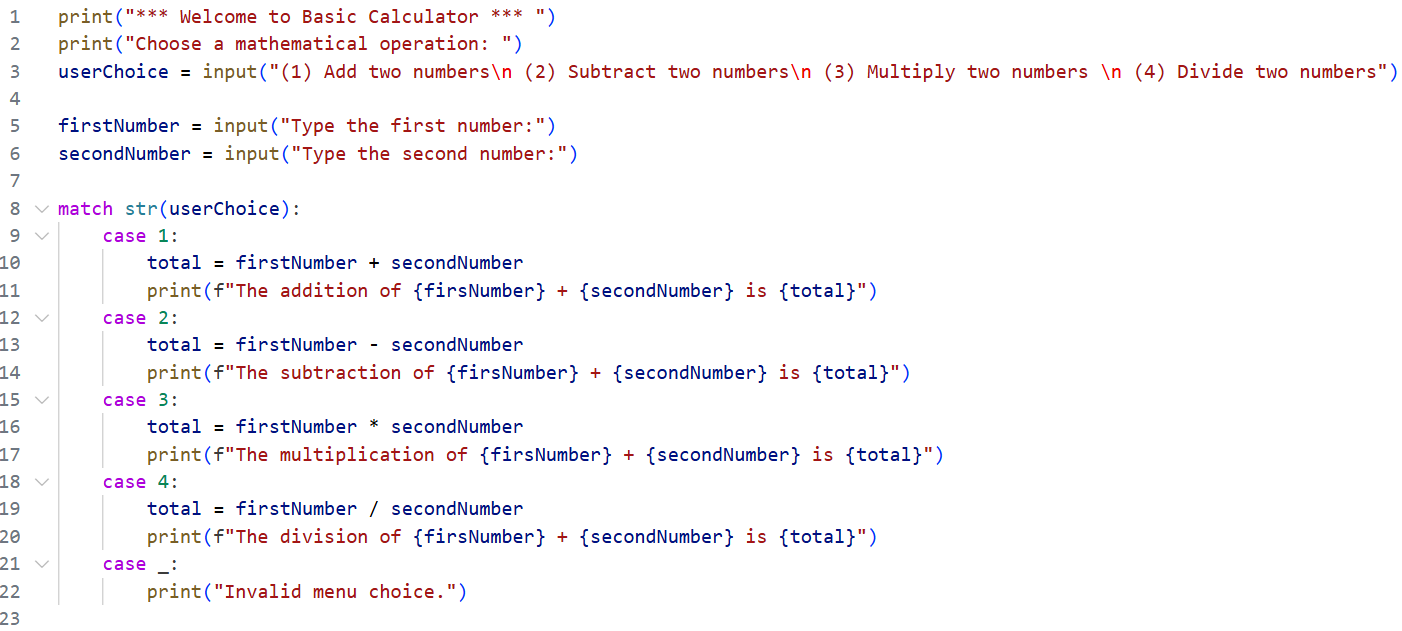


Figure 1 - Problematic Code Snippet for a basic calculator.

The purpose of this code is to create a simple calculator. Currently it wont work and has a lot of errors but the intent is clear. It will perform the 4 basic operations; Addition, Subtraction, Multiplication, and Division.

* **Task 2**: Implement this code in a file named bad\_calculator.py . As this name suggests, there is at least one problem with this code. Try the code out and identify all the problems you can with it.

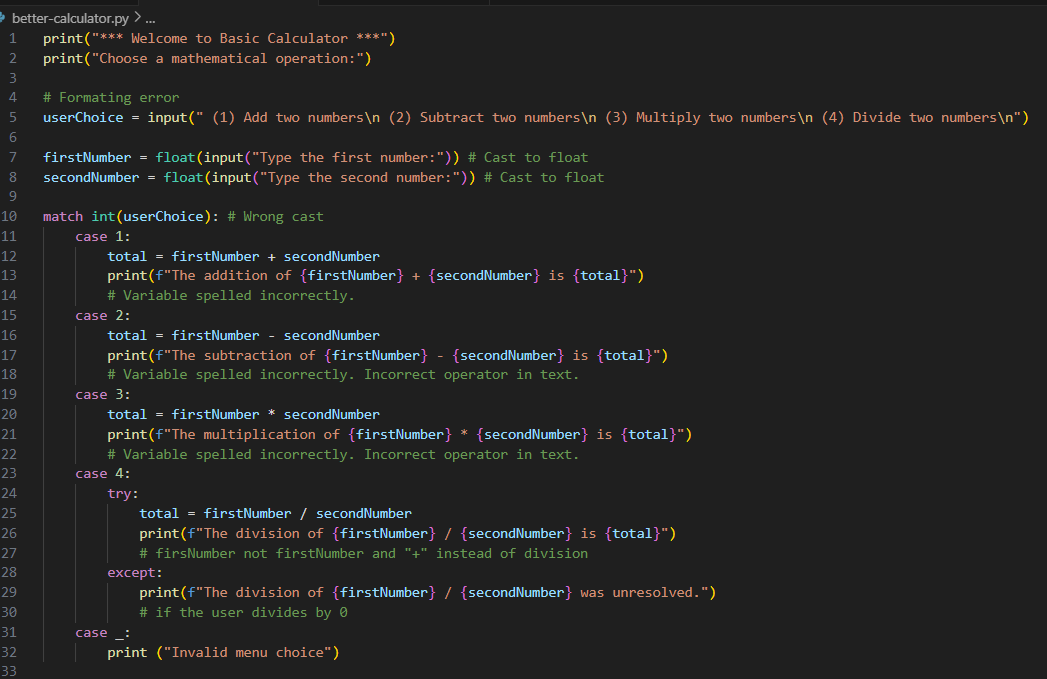
A screenshot of a computer program

AI-generated content may be incorrect.

I have written the errors I’ve found as comments in the text. But a TLDR would be that there’s a lot of casting issues that need to be fixed along with a lot of misspellings. It would also be smart if we could check if user choice was valid before we ask for input for firstNumber and secondNumber as they grow irrelevant otherwise.

If the second number is 0 the division at case 4 will throw an error. Nor are we double checking if the user input really is a float properly.

* **Task 3**: Create a file named better\_calculator.py where you fix the problems you identified in task 3.



Here I have implemented the fixes I showcased. It works with decimals for all of the operations.

* **Task 4**: Refactor your better\_calculator.py to have functions to perform the addition, subtraction, multiplication and division of two numbers passed as parameters. That way the code can be used by other programs in the future.

A screenshot of a computer program

AI-generated content may be incorrect.

I shifted how I call them too. Creating a function to call for inputs so I don’t have to get them when I get the operation.

A computer screen shot of a program code

AI-generated content may be incorrect.

I also made a function called “MathOperation” where I can pass a function and two inputs to do the operation. This allows me to easily update how I handle the results for all of them at once.

A screen shot of a computer program

AI-generated content may be incorrect.

Once done it looks like this. PS this code includes the solution to Task 5.

* **Task 5**: Add one more mathematical operation of your choice to better\_calculator.py.

A computer screen shot of numbers

AI-generated content may be incorrect.

I added floor division and modulus.

A screen shot of a computer code

AI-generated content may be incorrect.

* **Task 6**: Have you ever heard of the popular game “Jankenpo (じゃんけんぽん)” also known as “Rock-paper-scisors”? It is a two player game, where the player has to choose between using “rock”, "scissors" or “paper”. The rules are simple:
  + Rock beats Scissors
  + Scissors beats Paper
  + Paper beats Rock

The objective of this task is that you implement the code in a file named [**jankenpo.py**](http://jankenpo.py) to allow for a human player to play against a machine. **Hint:** One way you can randomize the action chosen by the computer is to have a dice.rollD3() function, and assign one action to each possible value.

A screenshot of a computer

AI-generated content may be incorrect.

(I lost) But I made a working code that lets you play Jankenpo against a bot.

Heres the main game loop:

A computer screen shot of text

AI-generated content may be incorrect.

MoveVersusMove calls into a massive nestled match statement that figures out if we win or not then calls functions depending on that. It looks like this:

A screen shot of a computer program

AI-generated content may be incorrect.

The picture only shows the logic for if the player picks rock, but essentially all moves follow in a similar manner.

Below is an image showcasing the functions helping MoveVersusMove to operate, all of them have a lot of fluff.

A screenshot of a computer program

AI-generated content may be incorrect.

The figure below shows the helper functions for converting moves between integers and strings for checks.

A screen shot of a computer program

AI-generated content may be incorrect.

* **Task 7**: Jankenpo is a really popular game where people add new variations (new hand movements with new names) all the time. Create your own variation by giving it a name and specific rules. Describe the name here and its specific rules. Implement this version in a file named [**myjankenpo.py**](http://jankenpo.py)

I made a new move called Carlos. Where you pull out your teacher Carlos that wins the round for you.

A screenshot of a computer program

AI-generated content may be incorrect.

The new MoveVersusMove logic:



How the game looks:

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.